Syntactical approaches to opetopes
Pierre-Louis Curien, Cédric Ho Thanh, Samuel Mimram

To cite this version:
hal-02064784
Syntactic approaches to opetopes

Pierre-Louis Curien
Cédric Ho Thanh
Samuel Mimram

Research Institute for Foundations of Computer Science (IRIF), Paris Diderot University, Paris, France
Email address: curien@irif.fr
URL: www.irif.fr/~curien

Research Institute for Foundations of Computer Science (IRIF), Paris Diderot University, Paris, France
Email address: cedric.hothanh@irif.fr
URL: hothanh.fr/cedric

Computer Science Laboratory of École Polytechnique (LIX), Palaiseau, France
Email address: samuel.mimram@lix.polytechnique.fr
URL: www.lix.polytechnique.fr/Labo/Samuel.Mimram

1This author has received funding from the European Union’s Horizon 2020 research and innovation program under the Marie Skłodowska-Curie grant agreement №665850.
Abstract. Opetopes are algebraic descriptions of shapes corresponding to compositions in higher dimensions. As such, they offer an approach to higher-dimensional algebraic structures, and in particular, to the definition of weak $\omega$-categories, which was the original motivation for their introduction by Baez and Dolan. They are classically defined inductively (as free operads in Leinster's approach, or as zoom complexes in the formalism of Kock et al.), using abstract constructions making them difficult to manipulate with a computer.

In this paper, we present two purely syntactic descriptions of opetopes as sequent calculi, the first using variables to implement the compositional nature of opetopes, the second using a calculus of higher addresses. We prove that well-typed sequents in both systems are in bijection with opetopes as defined in the more traditional approaches. Additionally, we propose three variants to describe opetopic sets. We expect that the resulting structures can serve as natural foundations for mechanized tools based on opetopes.
Contents

Chapter 1. Introduction 5
  1.1. Opetopes 5
  1.2. Generating opetopes 6
  1.3. Plan 8
  1.4. Related works 8

Chapter 2. Preliminaries 9
  2.1. Polynomial functors and trees 9
  2.2. Opetopes 10

Chapter 3. Named approach 15
  3.1. The system for opetopes 15
  3.2. Equivalence with polynomial opetopes 20
  3.3. Examples 26
  3.4. Python implementation 28
  3.5. The system for opetopic sets 29
  3.6. Equivalence with opetopic sets 31
  3.7. Examples 36
  3.8. Python implementation 38
  3.9. The mixed system for opetopic sets 39
  3.10. Equivalence with opetopic sets 41
  3.11. Examples 44
  3.12. Python implementation 46

Chapter 4. Unnamed approach 47
  4.1. The system for opetopes 47
  4.2. Equivalence with polynomial opetopes 52
  4.3. Examples 54
  4.4. Deciding opetopes 56
  4.5. Python implementation 57
  4.6. The system for opetopic sets 59
  4.7. Equivalence with opetopic sets 60
  4.8. Examples 62
  4.9. Python implementation 64

Chapter 5. Conclusion 67

Bibliography 69

Appendix A. Polynomial monads and the Baez–Dolan \((-)^+\) construction 71
  A.1. Polynomial monads 71
  A.2. A complete \((-)^+\) construction 73
CHAPTER 1

Introduction

1.1. Opetopes

Opetopes were originally introduced by Baez and Dolan in order to formulate a definition of weak ω-categories [Baez and Dolan, 1998]. Their name reflects the fact that they encode the possible shapes for higher-dimensional operations: they are operation polytopes. Over the recent years, they have been the subject of many investigations in order to provide good working definitions of opetopes allowing to explore their combinatorics [Cheng, 2003, Hermida et al., 2000, Leinster, 2004]. One of the most commonly used nowadays is the formulation based on polynomial functors and the corresponding graphical representation using “zoom complexes” [Kock et al., 2010].

In order to grasp quickly the nature of opetopes, consider a sequence of four composable arrows

\[
\begin{array}{cccc}
  a & \xrightarrow{f} & b & \xrightarrow{g} & c & \xrightarrow{h} & d & \xrightarrow{i} & e
\end{array}
\]

There are various ways we can compose them. For instance, we can compose \( f \) with \( g \), as well as \( h \) with \( i \), and then \( g \circ f \) with \( i \circ h \). Or we can compose \( f \), \( g \) and \( h \) together all at once, and then the result with \( i \). These two schemes for composing can respectively be pictured

\[
\begin{array}{c}
  f \\
  \downarrow \gamma \\
  a \\
  \vdots \\
  g \\
  \downarrow \beta \\
  b \\
  \vdots \\
  h \\
  \downarrow \delta \\
  c \\
  \vdots \\
  i \\
  \downarrow \epsilon \\
  e
\end{array}
\quad \text{and} \quad
\begin{array}{c}
  f \\
  \downarrow \gamma \\
  a \\
  \vdots \\
  g \\
  \downarrow \beta \\
  b \\
  \vdots \\
  h \\
  \downarrow \delta \\
  c \\
  \vdots \\
  i \\
  \downarrow \epsilon \\
  e
\end{array}
\]

From there, the general idea of getting “higher-dimensional” is that we should take these compositions as “2-operations”, which can be composed in various ways. For instance, in the first case, we can compose \( \alpha \) with \( \gamma \), and then \( \beta \) with the result, or all three at once, and so on. The opetopes describe all the ways in which these compositions can be meaningfully specified, in arbitrary dimension.

We can expect (and it is indeed the case) that the combinatorics of these objects is not easy to describe. In particular, a representation which is adapted to computer manipulations and proofs is desirable: we can for instance mention the Opetopic proof assistant for higher categories [Finster, 2016], which is based on opetopes. Recently, Mimram and Finster have used sequent calculus as a convenient tool to describe globular weak ω-categories [Finster and Mimram, 2017]. Our goal is to achieve a similar type-theoretic presentation for opetopic weak ω-categories [Baez and Dolan, 1998, Hermida et al., 2002], and we begin here by defining a representation of opetopes and opetopic sets of type-theoretic flavor.

Let us now informally define opetopes. At the basis of the architecture, there is a unique 0-opetope (i.e. opetope of dimension 0), drawn as a point. An (\( n + 1 \))-opetope (i.e. an opetope of dimension \( n + 1 \)) is made out of \( n \)-opetopes, and has a source and a target. In small dimensions, opetopes can be described using drawings of the kind above. For instance, the following are, from left to right, the unique 0-opetope, the unique 1-opetope, drawn as an arrow, and three 2-opetopes, respectively,
while the following drawing represents a 3-opetope:

In these drawings, the target of the 1-opetope is a point, the target of the 2-opetopes is the arrow at the bottom, and the target of the 3-opetope is the 2-opetope on the right, while the source of the 1-opetope is again a point, the source of the 2-opetopes is a diagram made of arrows, and the source of the 3-opetope is the diagram made of 2-opetopes on the left of the central arrow.

At this stage, we can already make the following observations:

1. each drawing contains only one top-dimensional cell, whose dimension determines the dimension of the opetope,
2. if an opetope has a non empty source, then its source and target have the same source (which might be empty) and the same target,
3. if an opetope has an empty source, then its target has the same source and target,
4. an \((n+1)\)-cell can have multiple \(n\)-cells in its source (including none) but always has exactly one \(n\)-cell in its target.

By the above remarks, an \(n\)-opetope has a unique top-dimensional cell \(\alpha\), whose target consists of one cell \(\beta\) whose source and target are the same as the ones of the source of \(\alpha\). It follows recursively that the opetope is entirely determined by the source of \(\alpha\), which we call the associated pasting scheme, which is of dimension \(n - 1\). The opetope associated to a pasting scheme of dimension \(n\) can be obtained by adding a single cell \(\beta\) of dimension \(n\) parallel to the pasting scheme, and an \((n+1)\)-cell \(\alpha\) from the pasting scheme to \(\beta\).

This article aims to provide syntactic tools making opetopes easier to manipulate than their classical definitions. The unnamed (or anonymous) approaches of chapter 4 on page 47 are purely relying on a calculus of higher addresses, and on a syntactic notion of preopetope. Opetopes are then well-formed preopetopes according to the derivation system \(\text{Opt}^2\) presented in figure 4.1.1 on page 50, while opetopic sets are derivable contexts in system \(\text{OptSet}^2\). The named approaches of chapter 3 on page 15, while slightly more complicated, leverage the idea of cell naming to produce user friendlier tools and results. As such it comes in two variants: \(\text{Opt}^1\) for describing opetopes, and \(\text{OptSet}^1\) for opetopic sets, introduced in figures 3.1.1 and 3.5.1 on page 17 and on page 30 respectively. The Python implementation of \cite{Ho Thanh, 2018b} is also discussed.

1.2. Generating opetopes

Our two opetope derivation systems \(\text{Opt}^1\) (figure 3.1.1 on page 17) and \(\text{Opt}^2\) (figure 4.1.1 on page 50) are based on the observation that opetopes are precisely all the shapes one can generate with the following operations.

1. Introduction of a point. There is a unique 0-opetope (the point).
   \[\text{point}\]

2. Shift to the next dimension. Given an \(n\)-opetope \(\omega\), we can form the \((n+1)\)-globe whose source and target are \(\omega\), as illustrated below. It can geometrically be thought of as the “extrusion” of \(\omega\).
   \[\text{shift}\]

3. Introduction of degeneracies. Given an \(n\)-opetope \(\omega\), we can build an \((n+2)\)-opetope with empty source, whose target is the globe at \(\omega\), as illustrated below for \(n = 0\) and \(n = 1\):
   \[\text{degen}\]
(4) **Grafting.** Given an \((n+1)\)-opetope \(\alpha\) and an \((n+1)\)-pasting scheme \(\beta\) such that the source of \(\beta\) contains an \(n\)-cell of the same shape as the target of \(\alpha\), we can graft \(\alpha\) to \(\beta\):

\[
\begin{array}{c}
\alpha \quad \text{graft} \quad \beta
\end{array}
\]

Ill-formed graftings may occur with \(n\)-pasting diagrams, for \(n \geq 3\), and the side condition is necessary to rule them out. Here is an example the graft rule will not allow: we deal with a 3-pasting diagram on the right of the dashed arrow (that comprises a unique 3-opetope), and the dashed arrow indicated that we attempt to graft the 3-opetope on the left (whose target shape is a trapezoid) onto the triangle shaped cell on the right:

\[
\begin{array}{c}
\alpha \quad \Rightarrow \quad \beta
\end{array}
\]

However, grafting onto the lower trapezoid of the right opetope is possible:

\[
\begin{array}{c}
\alpha \quad \Rightarrow \quad \beta
\end{array}
\]

As previously mentioned, an opetope is completely determined by its source pasting scheme, i.e. “arrangement” of source faces (the dichotomy between opetopes and pasting schemes is more thoroughly discussed in section 2.2.2 on page 11). We can reformulate rules shift and graft with this point of view to respectively obtain:

(1) **Filling of pasting schemes.** Given an \(n\)-pasting scheme, we may “fill” it by adding a target \(n\)-cell, and a top dimensional \((n+1)\)-cell. We illustrate an instance of this rule on the left, and invite the reader to compare it with the instance of shift on the right:

\[
\begin{array}{c}
\alpha \quad \text{fill} \quad \beta
\end{array}
\]

(2) **Substitution**, which consists in replacing a cell in a pasting scheme by another “parallel” pasting scheme. As before, we illustrate an instance if this rule on the left, and invite the reader to compare it with the instance of graft on the right:

\[
\begin{array}{c}
\alpha \quad \text{subst} \quad \beta
\end{array}
\]

In the case of opetopes, substitution can be illustrated as follows:

\[
\begin{array}{c}
\alpha \quad \text{subst}
\end{array}
\]
Rules point and degen can also be reformulated in term of pasting schemes, but representing them graphically is not as insightful.

1.3. Plan

In chapter 2 on the facing page, we expound some prerequisites and general ideas motivating our syntactic approaches. We then present the “unnamed” and “named” systems for opetopes and opetopic sets in chapters 3 and 4 on page 15 and on page 47 respectively. Those two chapters can be read independently, but have a similar structure:

1. firstly, we present the syntactic constructs and inference rules of systems Opt^1 (named) and Opt^7 (unnamed) for opetopes, in sections 3.1 and 4.1 respectively;
2. we then prove the equivalence with opetopes as defined in [Kock et al., 2010] (which we shall refer to as polynomial opetopes) in sections 3.2 and 4.2;
3. we showcase the Opt^1 and Opt^7 systems in examples in sections 3.3 and 4.3;
4. we discuss the Python implementation of Opt^1 and Opt^7 of [Ho Thanh, 2018b] in sections 3.4 and 4.5;
5. in the second part of those chapters, we present the syntactic constructs and inference rules of systems OptSet^1 (named) and OptSet^7 (unnamed) for opetopic sets, in sections 3.5 and 4.6 respectively;
6. we then prove the equivalence with opetopic sets of [Ho Thanh, 2018a] in sections 3.6 and 4.7;
7. we showcase the OptSet^1 and OptSet^7 systems in examples in sections 3.7 and 4.8;
8. we discuss the Python implementation of OptSet^1 and OptSet^7 sections 3.8 and 4.9;
9. for the named approach of chapter 3, we present the additional “mixed” system OptSet^1_m in section 3.9 on page 39, with the same plan as previously presented.

The equivalence proofs with polynomial opetopes of sections 3.2 and 4.2 on page 20 and on page 52 rely on a precise definition of the Baez–Dolan (−)∗ construction [Kock et al., 2010, Baez and Dolan, 1998], which is presented in appendix A on page 71.

1.4. Related works

Our syntactic opetopes are shown in sections 3.2 and 4.2 on page 20 and on page 52 to be equivalent to the polynomial opetopes (or “zoom complexes”) of Kock et al. [Kock et al., 2010], which are themselves equivalent to Leinster’s opetopes [Leinster, 2004]. It is known that the latter are incompatible with Cheng’s opetopes [Cheng, 2003], which should be thought of as being symmetric. There is a closely related notion of multitope [Hermida et al., 2002, Harnik et al., 2008] which is defined in terms of multicategories (whence the multi) instead of operads (ope); the two notions can be shown to be equivalent [Ho Thanh, 2018a]. A syntax for multitopes was proposed in [Hermida et al., 2002], where however not all the desired computations have been given algorithmic formulations.

The Opetopic proof assistant [Finster, 2016] for weak higher categories relies on the notion of higher-dimensional tree. In that system, the notion of opetope is built-in, so that we have to trust the implementation. In contrast, the present approach allows us to reason about the construction of opetopes. We moreover believe that the ability to reason by induction on the proof trees, together with the very explicit nature of our syntaxes, will allow for optimizations in the automated manipulations of opetopes.

Another proof assistant for weak higher categories, called CaTT [Finster and Mimram, 2017], starts from the same idea of generating well-formed pasting schemes through inference rules. However, it is based on globular shapes instead of opetopic ones, making a comparison with the present work difficult: since their introduction, people have unsuccessfully tried to compare the resulting respective categorical formalisms; we hope that their formulation in a common logical language might be of help in this task.

We should also mention here the Globular proof assistant [Bar et al., 2016], also based on globular shapes, which is quite popular, notably thanks to its nice graphical interface.
CHAPTER 2

Preliminaries

2.1. Polynomial functors and trees

We recall basic knowledge on the theory of polynomial functors from e.g. [Kock, 2011, Kock et al., 2010]. A polynomial endofunctor \( F \) over \( I \) is a \( \delta \)-set-diagram of the form:

\[
\begin{array}{ccc}
I & \overset{s}{\leftarrow} & E \\
& \overset{p}{\longrightarrow} & B \\
& \overset{t}{\longrightarrow} & I.
\end{array}
\]

(2.1.1)

Elements of \( B \) are called nodes or operations, elements of the fiber \( E(b) := p^{-1}(b) \) are the inputs or arities of \( b \), and elements of \( I \) are colors. For \( b \in B \), if \( e \in E(b) \), let \( s_e(b) := s(e) \). We will sometimes refer to \( I \), \( B \) and \( E \) as \( I_F \), \( B_F \), and \( E_F \), respectively. A morphism \( f : F \to F' \) of polynomial functors is a diagram of the form

\[
\begin{array}{ccc}
I_F & \overset{f}{\leftarrow} & E_F \\
& \overset{f}{\longrightarrow} & B_F \\
& \overset{f}{\longrightarrow} & I_F
\end{array}
\]

where all three squares commute, and the middle one is cartesian. This latter condition means that for \( b \in B_F \), the map \( f : E_F \to E_{F'} \) exhibits a bijection between the set \( E_F(b) \) of inputs of \( b \), and \( E_{F'}(f(b)) \) of inputs of \( f(b) \). We note \( \text{PolyEnd} \) the category of polynomial endofunctors and morphisms, and \( \text{PolyEnd}(I) \) the category of polynomial endofunctors over the set \( I \), where morphisms as above are required to satisfy \( f = \text{id}_I \).

A polynomial tree is a polynomial endofunctor \( T \) such that

1. \( I_T \) (also called the set of edges in this case), \( B_T \) (also called the set of nodes), and \( E_T \) are finite, and moreover \( I_T \neq \emptyset \).
2. \( s \) and \( t \) are injective, and the complement of the image of \( s \) has a unique element \( r \in I_T \) which we call the root edge,
3. for every edge \( i \in I_T \), there exists \( k \in \mathbb{N} \) such that \( s^k(i) = r \),

where the walk-to-root function \( \sigma : I_T \to I_T \) is inductively defined by \( \sigma(r) = r \), and, for \( i \neq r \in I_T \), \( \sigma(i) = tp(s^{-1}(i)) \). An edge that is not in the image of \( \sigma \) is called a leaf. Let \( \text{Tree} \) be the full subcategory of \( \text{PolyEnd} \) spanned by trees. We emphasize that morphisms in \( \text{Tree} \) are not required to be identities on edges, unlike what would happen if \( \text{Tree} \) was a subcategory of \( \text{PolyEnd}(I) \).

For \( F \in \text{PolyEnd} \), let \( \text{tr} \ F \), the category of \( F \)-trees, be a chosen skeleton of \( \text{Tree}/F \). Depending on the context, it may also refer to the set of isomorphism classes of \( F \)-trees. An \( F \)-tree is thus a tree \( (T) \) and a morphism of polynomial functors \( T : (T) \to F \). For \( x \) an edge of \( T \) (or more precisely of \( (T) \)), the morphism \( T \) associates a color \( i = T(x) \in I_F \), and we say that \( x \) is decorated with \( i \), and likewise for nodes.

Let \( T \in \text{tr} \ F \), and \( \sigma \) the walk-to-root function of \( (T) \). Write \( (T) \) as

\[
I_T \overset{s}{\leftarrow} E_T \overset{p}{\longrightarrow} B_T \overset{t}{\longrightarrow} I_T.
\]

For an edge \( x \in I_T \) we define inductively the address of \( x \), noted \&\( x \), which is a bracket-enclosed list of elements of \( E_T \), indicating the path from the root to that edge. If \( x \) is the root edge, then \&\( x = [] \), the empty list. Otherwise, let \&\( x = (\&\sigma(x)) \cdot [x] \), where \( \cdot \) is the concatenation of lists. If \( y \in B_T \) is a node of \( T \), let its address be given by \&\( y = \&t(y) \), that is, it is the address of its output edge. Write \( T^* \) the set of node addresses of \( T \), and for \( [p] \in T^* \), let \( s_{[p]}T = T(y) \), where \( y \in B_T \) is the node such that \&\( y = [p] \). In other words \( s_{[p]}T \in B_F \) is the decoration of the node of \( T \) at address \( [p] \). We let \( T^l \) be the set of leaf addresses of \( T \).
Let $F$ be a polynomial endofunctor as in equation (2.1.1). For $i \in I$, define $I_i \in \text{tr } F$ as having underlying tree

\[
\{ i \} \leftrightarrow \emptyset \rightarrow \emptyset \rightarrow \{ i \},
\]

and the obvious morphism to $F$. This corresponds uniquely to the $F$-tree with no node and a unique edge, decorated by $i$. Equivalently, it corresponds uniquely to the edge address of an edge decorated in $i$. Let now $b \in B$, and define $Y_b \in \text{tr } F$, the corolla at $b$, as having underlying tree

\[
E(b) + \{ t(b) \} \leftarrow E(b) \rightarrow \{ b \} \rightarrow E(b) + \{ t(b) \},
\]

where the left map is increasing with codomain $\{1, \ldots, n\}$, and the right one maps $b$ to $n + 1$. This corresponds to a $F$-tree with a unique node, decorated by $b$, or equivalently, to the node address of such a node.

Let $S, T \in \text{tr } F$ be two $F$-trees, where the root edge (that at address []) of $T$ is decorated by $i \in I_F$, and let $[l] \in S$ be the address of a leaf of $S$ decorated by $i$. We define the grafting $S \circ_{[l]} T$ of $S$ and $T$ on $l$ by the following pushout (in $\text{tr } F$):

\[
\begin{array}{ccc}
I_i & \xrightarrow{r} & T \\
\downarrow & & \downarrow \\
S & \xrightarrow{r} & S \circ_{[l]} T.
\end{array}
\]

**Proposition 2.1.5** ([Kock, 2011]). Every $F$-tree is either of the form $I_i$, for some $i \in I$, or obtained by graftings a corollas on an $F$-tree.

Take $T, U_1, \ldots, U_k \in \text{tr } F$, where the leaves of $T$ have addresses $[l_1], \ldots, [l_k]$, and assume the grafting $T \circ_{[l_i]} U_i$ is defined for all $i$. Then the total grafting will be denoted concisely by

\[
T \bigcirc_i U_i = (\cdots (T \circ_{[l_1]} U_1) \circ_{[l_2]} U_2 \cdots) \circ_{[l_k]} U_k.
\]

It is easy to see that the result does not depend on the order in which the graftings are performed.

2.1.1. **The polynomial Baez–Dolan $(-)^*$ construction.** We now give a brief definition of the $(-)^*$ construction, see appendix A on page 71 for details. A polynomial monad $(M, \mu, \eta)$ is simply a monad in the 2-category $\text{PolyEnd}$. This structure induces a function $\tau : \text{tr } M \rightarrow B_M$, for each $T \in \text{tr } M$ a bijection $\varphi_T : T^! \rightarrow (tT)^*$, called readdressing, see appendix A.1 on page 71.

From such a monad $M$, we may define a new polynomial monad $M^+$ whose underlying functor is given by

\[
\begin{array}{ccc}
B_M & \xleftarrow{\varsigma} & \text{tr}^* M \\
\uparrow & & \uparrow \\
\uparrow & & \uparrow \\
\text{tr } M & \xrightarrow{u} & \text{tr } M \xrightarrow{t} B_M
\end{array}
\]

where $\text{tr}^* M$ is the set of $M$-trees with one marked node (or equivalently, equipped with one of their node addresses), $\varsigma$ gives the marked node (or more precisely its decoration), $u$ forgets the marking, and $t$ is induced from the monad law of $M$. The monad structure of $M^+$ is given by substitution of trees, see appendix A.2 on page 73.

2.2. **Opetopes**

2.2.1. **Definition.** Let $3^0$ be the identity polynomial monad on $\mathbb{S}$, on the left, and define $3^{n+1} := (3^n)^*$.

\[
\{ * \} \leftarrow \{ * \} \rightarrow \{ * \} \rightarrow \{ * \}, \quad \mathbb{O}_n \xleftarrow{\varsigma} E_3^{n+1} \rightarrow \mathbb{O}_{n+1} \xrightarrow{t} \mathbb{O}_n.
\]

If we write the polynomial functor $3^n$ as on the right, an $n$-opetope $\omega$ is by definition an element of $\mathbb{O}_n$, and we say that $\omega$ has dimension $n$, written $\text{dim } \omega = n$. The map $\varsigma$ is called the source map, and $t$ the target map.

The unique 0-opetope is written $\bullet$ and called the point, whereas the unique 1-opetope is written $\bullet$ and called the arrow. For $n \geq 2$, we have $\mathbb{O}_n = \text{tr } 3^{n-2}$, hence an $n$-opetope is a tree whose nodes and edges are decorated with $(n-1)$-opetopes and $(n-2)$-opetopes, respectively. For $\omega \in \mathbb{O}_n$, the set $E_3^n(\omega)$ is precisely $\omega^*$, i.e. is the set of node addresses of $\omega$. The readdressing of $\omega$ is then of the form $p_\omega : \omega^! \rightarrow (t\omega)^*$.
2.2. Opetopes vs. pasting schemes. Opetopes are closely related to the notion of “pasting scheme” commonly used in higher category theory to describe composition of higher cells. Informally, a pasting scheme of dimension \( n \) is a tree whose nodes are decorated with \( n \)-“cells”, edges with \((n-1)\)-cells, and where the output edge of a node corresponds to its target, and the input edges to the cells in its source. For instance, the figure on the left represents a 2-pasting scheme, and the corresponding tree is drawn on the right:

![Diagram of a 2-pasting scheme]

If we consider the \( k \)-cells as \( k \)-opetopes for all \( k \), then an \( n \)-pasting scheme \( P \) induces a \( \mathcal{3}^{n-1} \)-tree, thus an \((n+1)\)-opetope, say \( \omega \). We say that \( P \) is the source pasting scheme of \( \omega \). Further, by definition of \( \mathcal{3}^n \), the opetope \( \omega \) has a target \( ! \omega \in \Omega_{n-1} \). In the sequel, graphical representations of opetopes include both the source pasting scheme and the target: for instance, if \( P \) is the pasting scheme above, then \( \omega \) is represented by

![Diagram of an opetope]

Such a graphical representation is ambiguous however: the following

![Diagram of a 2-opetope]

represents a 2-opetope (whose underlying 1-pasting scheme has three 1-cells), but also a 2-pasting scheme having a unique 2-cell. In this work, such ambiguities shall be lifted by the surrounding context.

2.2.3. Higher addresses. We now introduce the concept of higher dimensional address, a refinement if that of addresses presented in section 2.1 on page 9, which will serve as a more convenient way to designate nodes and edges of opetopes. Start by defining the set \( \mathcal{A}_n \) of \( n \)-addresses as follows:

\[
\mathcal{A}_0 = \{ * \}, \quad \mathcal{A}_{n+1} = \text{lists} \mathcal{A}_n,
\]

where lists \( X \) is the set of finite lists (or words) on the alphabet \( X \). Explicitly, the unique 0-address is \(*\) (also written \([\ ]\) by convention), while an \((n+1)\)-address is a sequence of \( n \)-addresses. Such sequences are enclosed by brackets. Note that the address \([\ ]\) associated to the empty word, is in \( \mathcal{A}_n \) for all \( n \geq 0 \). However, the surrounding context will almost always make the notation unambiguous\(^1\). Here are examples of the words we shall use:

\[
[\ ] \in \mathcal{A}_1, \quad [\ast \ast \ast ] \in \mathcal{A}_1, \quad [[[\ast ]]]) \in \mathcal{A}_2, \quad [[[\ast ]]]) \in \mathcal{A}_4 \quad \ldots
\]

For \( \omega \in \Omega \) an opetope, nodes of \( \omega \) can be uniquely specified using higher addresses as we now show. In \( \mathcal{3}^0 \), set \( E_3(\bullet) = \{ \ast \} \), so that the unique node address of \( \bullet \) is \( * \in \mathcal{A}_0 \). For \( n \geq 2 \), recall that an opetope \( \omega \in \Omega_n \) is a \( \mathcal{3}^{n-2} \)-tree:

\[
\omega : \langle \omega \rangle \rightarrow \mathcal{3}^{n-2}.
\]

A node \( b \in B_\omega \) has an address \& \( b \) lists \( E_\omega \), which by \( \omega : E_\omega \rightarrow E_{3^{n-1}} \) is mapped to an element of lists \( E_{3^{n-1}} \). By induction, elements of \( E_{3^{n-1}} \) are \((n-2)\)-addresses, whence \( E_\omega (\& b) \) is list \( \mathcal{A}_{n-2} = \mathcal{A}_{n-1} \). For the induction step, elements of \( E_{3^n}(\omega) \) are nodes of \( \langle \omega \rangle \), which we identify by their aforementioned \((n-1)\)-addresses. Consequently, for all \( n \geq 1 \) and \( \omega \in \Omega_n \), elements of \( E_{3^n}(\omega) = \omega^* \) can be seen as the set of \((n-1)\)-addresses of the nodes of \( \omega \), and similarly, \( \omega^\dagger \) can be seen as the set of \((n-1)\)-addresses of edges of \( \omega \).

We denote the concatenation of higher addresses by \( \cdot \), i.e. if \( [p_1], [p_2] \) are \( n \)-addresses, then \([p_1] \cdot [p_2] : = [p_1 p_2] \). Let \( \subseteq \) be the prefix order on the set of \( n \)-addresses: \( [p_1] \subseteq [p_2] \) if and only if as sequences, \( p_1 \) is a prefix of \( p_2 \). Let

\(^1\)Ambiguity with addresses could be lifted altogether by hinting the dimension as e.g. \([\ ]^1 \in \mathcal{A}_1, [\ast \ast \ast ]^1 \in \mathcal{A}_1, [[[\ast ]]^1]^1 \in \mathcal{A}_2, [[[\ast ]^1]^1]^1 \in \mathcal{A}_4 \). This however makes notations significantly heavier, so we avoid using this convention.
≤ be the lexicographical order on the set of \( n \)-addresses. Explicitly, it is trivial on \( A_0 \), given by the prefix order \( \preceq \) on \( A_1 \), and on \( A_n \), seen as the set of finite sequences of elements of \( A_{n-1} \), it is induced from the lexicographical order on \( A_{n-1} \). Remark that \( \preceq \) is always a suborder of \( \leq \).

**Example 2.2.1.** Consider the following 2-opetope, called 3:

![Diagram](image)

Its underlying pasting diagram consists of 3 arrows \( \bullet \) grafted linearly. Since the only node address of \( \bullet \) is \( * \in A_0 \), the underlying tree of 3 can be depicted as follows:

```
  *  
 / 
*    
```

On the left are the decorations: nodes are decorated with \( * \in O_1 \), while the edges are decorated with \( \bullet \in O_0 \). For each node int the tree, the set of input edges of that node is in bijective correspondence with the node addresses of the decorating opetope, and that address is written on the right of each edges. In this low dimensional example, those addresses can only be \( * \). Finally, on the right of each node is its 1-address, which is just a sequence of 0-addresses giving “walking instructions” to get from the root to that node.

The 2-opetope 3 can then be seen as a corolla in some 3-opetope as follows:

![Diagram](image)

As previously mentioned, the set of input edges is in bijective correspondence with the set of node addresses of 3. Here is now an example of a 3-opetope, with its annotated underlying tree on the right (the 2-opetopes 1 and 2 are analogous to 3):

![Diagram](image)

Let \( [l] = [p[q]] \in A_{n-1} \) be an address such that \( [p] \in \omega \) and \( [q] \in (s[p]) \omega \). Then as a shorthand, we write

\[
e_{[l] \omega} := s[q] s[p] \omega .
\]

(2.2.2)

The source map \( s \) give the decoration of a node in a tree, and likewise, \( e \) gives the decoration of edges.

**Remark 2.2.3** (Opetopic induction). The proposition 2.1.5 on page 10 allows a fundamental type of induction on opetopes that we use frequently throughout this paper. Let \( \Phi(\cdot) \) be a logical proposition about opetopes such that

1. \( \Phi \) holds for \( \bullet \), written \( \Phi(\bullet) \),
(2) \( \Phi(\psi) \equiv \Phi(1_\psi) \land \Phi(Y_\psi) \), for all \( \psi \in \mathcal{O} \),
(3) \( \Phi(\nu) \land \Phi(\psi) \equiv \Phi(\nu \circ \iota \psi) \), for \( \nu, \psi \in \mathcal{O} \) with at least one node, \([I] \in \nu\), and assuming the grafting is well-defined.

Then \( \Phi \) holds for all opetope.

2.2.4. The category of opetopes. In this subsection, we define the category \( \mathcal{O} \) of opetopes. We first begin by a result whose geometrical meaning is explained later in this section.

Lemma 2.2.4 (Opetopic identities, [Ho Thanh, 2018a]). Let \( \omega \in \mathcal{O}_n \) with \( n \geq 2 \).
(1) (Inner edge) For \([p[q]] \in \omega^*\) (forcing \( \omega \) to be non degenerate), we have \( ts[p[q]] \omega = s[q]s[p] \omega \).
(2) (Globularity 1) If \( \omega \) is non degenerate, we have \( ts[l] \omega = tt \omega \).
(3) (Globularity 2) If \( \omega \) is non degenerate, and \([p[q]] \in \omega^1\), we have \( s[q]s[p] \omega = s[p[q]]t \omega \).
(4) (Degeneracy) If \( \omega \) is degenerate, we have \( s[l] t \omega = tt \omega \).

With those identities in mind, we define the category \( \mathcal{O} \) of opetopes by generators and relations as follows.
(1) (Objects) We set \( \text{ob} \mathcal{O} = \sum_{n \in \mathbb{N}} \mathcal{O}_n \).
(2) (Generators) Let \( \omega \in \mathcal{O}_n \) with \( n \geq 1 \). We introduce a generator, called target embedding: \( t : t \omega \rightarrow \omega \). If \([p] \in \omega^*\), then we introduce a generator, called source embedding: \( s[p] : s[p] \omega \rightarrow \omega \). A face embedding is either a source or target embedding.
(3) (Relations) We impose 4 relations described by the following commutative squares, that are well defined thanks to theorem 2.2.4. Let \( \omega \in \mathcal{O}_n \) with \( n \geq 2 \)
(a) (Inner) for \([p[q]] \in \omega^*\) (forcing \( \omega \) to be non degenerate), the following square must commute:

\[
\begin{array}{ccc}
  s[q]s[p] \omega & \rightarrow & s[p] \omega \\
  \downarrow & & \downarrow \\
  s[p[q]] \omega & \rightarrow & \omega \\
\end{array}
\]

(b) (Glob1) if \( \omega \) is non degenerate, the following square must commute:

\[
\begin{array}{ccc}
  t t \omega & \rightarrow & t \omega \\
  t & \downarrow & t \\
  s[l] \omega & \rightarrow & \omega .
\end{array}
\]

(c) (Glob2) if \( \omega \) is non degenerate, and for \([p[q]] \in \omega^1\), the following square must commute:

\[
\begin{array}{ccc}
  s[p[q]] t \omega & \rightarrow & t \omega \\
  s[q] & \downarrow & t \\
  s[p] \omega & \rightarrow & \omega .
\end{array}
\]

(d) (Degen) if \( \omega \) is degenerate, the following square must commute:

\[
\begin{array}{ccc}
  t t \omega & \rightarrow & t \omega \\
  s[l] & \downarrow & t \\
  t \omega & \rightarrow & \omega .
\end{array}
\]

An opetopic set is a Set-valued presheaf over \( \mathcal{O} \). We write \( \mathcal{O} \) for the category of opetopic sets and natural transformations, \( O[-] : \mathcal{O} \rightarrow \mathcal{O} \) for the Yoneda embedding, and \( \text{Fin} \mathcal{O} \) for the full subcategory of \( \mathcal{O} \) spanned by finite opetopic sets, i.e. those \( X \in \mathcal{O} \) such that \( \sum_{\omega \in \mathcal{O}} X_\omega \) is a finite set. Equivalently, \( \text{Fin} \mathcal{O} \) is the completion of \( \mathcal{O} \) under finite colimits.

Let us explain this definition a little more. As previously mentioned, opetopes are trees whose nodes (and edges) are decorated by opetopes. The decoration is now interpreted as a geometrical feature, namely as an
embedding of a lower dimensional opetope. Further, the target of an opetope, while not an intrinsic data, is also represented as an embedding. The relations can be understood as follows.

1. **(Inner)** The inner edge at \([p,q]\) is decorated by the target of the decoration of the node “above” it (here \(s_{[p]}\)) and in the \([q]\)-source of the node “below” it (here \(s_{[p]}\)). By construction, those two decorations match, and this relation makes the two corresponding embeddings match as well. On the left is an informal diagram about \(\omega\) as a tree (reversed gray triangle), and on the right is an example of pasting diagram represented by an opetope, with the relevant features of the **(Inner)** relation colored or thickened.

2. **(Glob1-2)** If we consider the underlying tree of \(\omega\) (which really is \(\omega\) itself) as its “geometrical source”, and the corolla \(Y_t\) as its “geometrical target”, then they should be parallel. The relation **(Glob1)** expresses this idea by “gluing” the root edges of \(\omega\) and \(Y_t\) together, while **(Glob2)** glues the leaves according to the redressing function \(p_{\omega}\).

3. **(Degen)** If \(\omega\) is a degenerate opetope, depicted as on the right, then its target should be a “loop”, i.e. its only source and target should be glued together.
CHAPTER 3

Named approach

3.1. THE SYSTEM FOR OPETOPEs

3.1.1. Syntax. In this section, we define the underlying syntax of Opt¹, our named derivation system for opetopes. As explained in the introduction, a typical pasting scheme is pictured below:

We shall use the names of the cells of this picture as variables, and encode the pasting scheme as the following expression:

\[ (j \leftarrow \alpha, k \leftarrow \beta) \]

Here, \( j, k, \alpha, \beta \), and \( \gamma \) are now variables, equipped with a dimension (1 for \( j \) and \( k \), and 2 for \( \alpha, \beta, \) and \( \gamma \)), and the notation is meant to be read as “the variable \( \gamma \) in which \( \alpha \) (resp. \( \beta \)) has been formally grafted on the input labeled \( j \) (resp. \( k \))”. Such a term will be given a type:

\[ i(t \leftarrow h(z \leftarrow g(y \leftarrow f))) \rightarrow a \rightarrow \emptyset, \]

which expresses the fact that the source is the “composite” \( i \circ h \circ g \circ f \), and that the source of the source is \( x \). Since the pasting scheme is 2-dimensional, there is no further iterated source, and we conclude the sequence by a \( \emptyset \) symbol (which can be read as the only \((-1)\)-dimensional pasting scheme). Similarly, the degenerate pasting scheme on the left below will be denoted by the typed term figured on the right:

\[ f \]

\[ \alpha : x \rightarrow x \rightarrow \emptyset \]

where the term \( x \) denotes a degenerate 1-dimensional pasting scheme with \( x \) as source.

3.1.1.1. Terms. We have a \( \mathbb{N} \)-graded set \( V \) of variables. Elements of \( V_n \) represent \( n \)-dimensional cells. An \( n \)-term is constructed according to the following grammar:

\[
\begin{align*}
T_{-1} & ::= \{ \emptyset \} & \text{by convention} \\
T_0 & ::= V_0 \\
T_{n+1} & ::= V_n \mid T'_{n+1} \\
T'_{n+1} & ::= V_{n+1}(V_n \leftarrow T'_{n+1}, \ldots)
\end{align*}
\]

where the expression \( (V_n \leftarrow T'_{n+1}, \ldots) \) signifies that there is 0 or more instances of the “\( V_n \leftarrow T'_{n+1} \)” part between the parentheses. For example, if \( f, g \in V_1 \), and \( a \in V_0 \), then the following is an element of \( T_1 \):

\[ g(a \leftarrow f()) \]

To make notations lighter, we do not write empty parentheses “()”, so the previous 1-term can be more concisely written as \( g(a \leftarrow f) \). The terms of the form \( \emptyset \) are called degenerate terms, or empty syntactic pasting schemes.

A term of the form \( g(a_1 \leftarrow f_1, \ldots, a_k \leftarrow f_k) \) will oftentimes be abbreviated as \( g(a_i \leftarrow f_i) \), leaving \( k \) implicit. By convention, the sequence \( a_1 \leftarrow f_1, \ldots, a_k \leftarrow f_k \) above is always considered up to permutation: for \( \sigma \) a bijection of the set \( \{1, \ldots, k\} \), the terms \( g(a_1 \leftarrow f_1, \ldots, a_k \leftarrow f_k) \) and \( g(a_{\sigma(1)} \leftarrow f_{\sigma(1)}, \ldots, a_{\sigma(k)} \leftarrow f_{\sigma(k)}) \) are considered equal. Note that the above definition entails that \( T_0 = V_0 \) (in particular, there are no degenerate 0-terms).
For $t \in \mathbb{T}_n$, write $t^*$ for the set of $n$-variables occurring in $t$. In the previous example, $(g(a \leftarrow f))^* = \{f, g\}$. Note that $x \in x^*$ for all $x \in \mathbb{V}_n$. We consider terms of the form $y$, with $y \in \mathbb{V}_k$, as empty, or degenerate, $(n+1)$-terms, i.e. with no $(n+1)$-variables.

3.1.2. Types. An $n$-type $T$ is a sequence of terms of the form

$$s_1 \bowtie s_2 \bowtie \cdots \bowtie s_n \bowtie \emptyset,$$

where $s_i \in \mathbb{T}_{n-i}$. As we will see (rule shift in figure 3.1.1 on the next page, and theorem 3.1.8 on page 19), this sequence of terms essentially describes a zoom complex in the sense of [Kock et al., 2010], which justifies the use of the $\bowtie$ symbol. A typing of a term $t \in \mathbb{T}_n$ is an expression of the form $t : T$, for $T$ an $n$-type. If $T$ is as above, then $s_i$ is thought of as the $i$-th (iterated) source of $t$. We then write $s t := s_1$, and more generally $s^i t := s_i$.

By convention, $s^0 t := t$.

3.1.3. Contexts. A context $\Gamma$ is a set of typings, more commonly written as a list. Write $\mathbb{V}_{\Gamma, k}$ for the set of $k$-variables typed in $\Gamma$, let $\mathbb{V}_{\Gamma} := \sum_{k \in \mathbb{N}} \mathbb{V}_{\Gamma, k}$, write $\mathbb{T}_{\Gamma, k}$ for the set of $k$-terms whose variables (in any dimension) are in $\mathbb{V}_{\Gamma}$, and $\mathbb{T}_\Gamma := \sum_{k \in \mathbb{N}} \mathbb{T}_{\Gamma, k}$. As we will see (inference rules in figure 3.1.1 on the next page), for a derivable context $\Gamma$, if $x$ occurs in the typing of a variable of $\Gamma$, then $x \in \mathbb{V}_{\Gamma}$. Note that in any context $\Gamma$, if a variable $x \in \mathbb{V}_{\Gamma, k}$ occurs in the type of $y \in \mathbb{V}_{\Gamma, l}$, then $k < l$, and thus there is no cyclic dependency among variables.

3.1.4. Sequents. Let $\Gamma$ be a context. An equational theory $E$ on $\mathbb{V}_{\Gamma}$ is a set of formal equalities between variables of $\Gamma$. We write $=_{E}$ for the equivalence relation on $\mathbb{V}_{\Gamma}$ generated by $E$. A sequent is an expression of the form

$$E \vdash \Gamma \vdash t : T$$

where $\Gamma$ is a context, $E$ is an equational theory on $\mathbb{V}_{\Gamma}$, and the right hand side is a typing. We may write $\vdash_n$ to signify that $t \in \mathbb{T}_n$. The equivalence relation $=_{E}$ on $\mathbb{V}_{\Gamma}$ extends to $\mathbb{T}_{\Gamma}$ in an obvious way. If $x =_{E} y \in t^*$, then, by convention $x \in t^*$, so that $x$ and $y$ really are interchangeable.

If $(F \vdash Y \vdash u : U)$ is a sequent such that there exists a bijection $\sigma : \mathbb{V}_{\Gamma} \rightarrow \mathbb{V}_{\Gamma}$ with

$$(E \vdash \Gamma \vdash t : T) = (F^\sigma \vdash \Gamma^\sigma \vdash u^\sigma : U^\sigma),$$

where $(-)^\sigma$ is the substitution according to $\sigma$, then we say that both sequents are equivalent (or $\alpha$-equivalent), denoted

$$(E \vdash \Gamma \vdash t : T) \simeq (F \vdash \Gamma \vdash u : U).$$

In the following, sequents are implicitly considered up to equivalence.

3.1.2. Inference rules. We present the inference rules of the $\text{Opt}^\top$ system in figure 3.1.1 on the facing page. Rule graft requires the so-called graft notation and substitution operation, respectively introduced in definitions 3.1.1 and 3.1.3 on the next page.
Introduction of points: This rule introduces 0-cells, also called points. If $x \in \mathbb{V}_0$, then
$$\Gamma \vdash x : \emptyset \quad \text{point}$$

Introduction of degeneracies: This rule derives empty pasting diagrams. If $x \in \mathbb{V}_n$, then
$$\Gamma \vdash E \vdash \Gamma \vdash x : T \quad \text{degen}$$

Shift to the next dimension: This rule takes a term $t$ and introduces a new cell $x$ having $t$ as source. If $x \in \mathbb{V}_{n+1}$ is such that $x \notin \mathbb{V}_n$, then
$$\Gamma \vdash E \vdash \Gamma \vdash x : T \quad \text{shift}$$

Grafting: This rule glues an $n$-cell $x$ onto an $n$-term $t$ along a variable $a \in s_1 := (st)^*$. We assume that $\Gamma$ and $\Upsilon$ are compatible, in that for all $y \in \mathbb{V}$, if $y \in \mathbb{V}_\Gamma \cap \mathbb{V}_\Upsilon$, then the typing of $y$ in both contexts match modulo the equational theory $E \cup F$. Further, the only variables typed in both $\Gamma$ and $\Upsilon$ are $a$ and the variables occurring in the sources of $a$ (i.e. $s_1$, for $1 \leq i \leq n-1$).

If $x \in \mathbb{V}_n$, $t \in \mathbb{T}_n$ is not degenerate, $a \in (st)^*$ is such that $s_0 = ssx$, then
$$\Gamma \vdash E \vdash \Gamma \vdash t : s_1 \hookrightarrow s_2 \hookrightarrow \ldots \cdot F \vdash \Upsilon \vdash x : U \quad \text{graft}$$

where the notations $t(a \leftarrow x)$ and $s_1[sx/a]$ are presented below, where $G$ is the union of $E$, $F$, and potentially a set of additional equalities incurred by the substitution $s_1[sx/a]$. We also write $\text{graft-}a$ to make explicit that we grafted onto $a$.

The condition $a \in (st)^*$ ensures that $a$ hasn’t been used for grafting beforehand, while the condition $sa = ssx$ shows that $x$ may indeed be glued onto $a$.

**Definition 3.1.1** (Graft notation). For a sequent $(E \vdash \Gamma \vdash t : T)$, $a \in \mathbb{V}_{n-1}$, and $x \in \mathbb{V}_{\Upsilon,n}$, the graft notation $t(a \leftarrow x)$ of the graft rule can be simplified depending on the structure of $t$, according to the following rewriting rule: for $y \in \mathbb{V}_{\Gamma,n}$:
$$y(\overline{z_i} \leftarrow v_i)(a \leftarrow x) \quad \rightsquigarrow \quad \begin{cases} y(\overline{z_i} \leftarrow v_i(a \leftarrow x)) & \text{if } a \notin (sy)^*, \\ y(\overline{z_i} \leftarrow v_i, a \leftarrow x) & \text{if } a \in (sy)^*. \end{cases} \quad (3.1.2)$$

In particular, note that if $a \notin (sy)^*$, then $y() (a \leftarrow x) \rightsquigarrow y()$, and with the “empty parentheses convention”, this gives $y(a \leftarrow x) \rightsquigarrow y$.

**Definition 3.1.3** (Substitution). We now explain how to evaluate $u[w/a]$ for any term $u \in \mathbb{T}_{n-1}$. We let
$$u[w/a] := \begin{cases} y(\overline{z_i} \leftarrow v_i(w/a)) & \text{if } a \notin E \cup F \ y, \\ w(\overline{z_i} \leftarrow v_i) & \text{if } a \in E \cup F \ y. \end{cases}$$

and then, in the case $w = b$, we perform the following actions:

1. for subterms of the form $x(\ldots, z \leftarrow b, \ldots)$, we remove the grafting $z \leftarrow b$, and add the equation $b = z$ to the ambient equational theory;
2. for subterms of the form $x(\ldots, z \leftarrow b(b \leftarrow r), \ldots)$, we replace the grafting $z \leftarrow b(b \leftarrow r)$ by $z \leftarrow r$ and add $b = z$ to the ambient equational theory;
3. and finally replace any remaining subterms of the form $b(b \leftarrow v)$ by $v$.

More explicitly:

1. If $w$ is not an empty pasting diagram (i.e. not of the form $b$), then writing $u = y(\overline{z_i} \leftarrow v_i)$:
$$u[w/a] := \begin{cases} y(\overline{z_i} \leftarrow v_i(w/a)) & \text{if } a \notin E \cup F \ y, \\ w(\overline{z_i} \leftarrow v_i) & \text{if } a \in E \cup F \ y. \end{cases} \quad (3.1.4)$$

\[\text{Figure 3.1.1. The Opt}^1 \text{ system.}\]
(2) If \( w \) is an empty pasting diagram, say \( w = \emptyset \) for \( b \in \mathbb{V}_{n-2} \), then, by the hypothesis of the \textbf{graft} rule, we have \( b \models_{E,F} a \). Then \( u[b/a] \) is defined by cases on the form of \( u \):

- if \( u = E \cup F \) a, then \( u[b/a] = b \);
- if \( u \) is of the form \( a(b \leftarrow r) \), then \( u[b/a] = r \);
- if \( u \) is of the form \( y(\ldots, z \leftarrow a, \ldots) \), then
  \[
  u[b/a] = y(\ldots, \ldots),
  \]
  and we add the equality \( b = z \) to the ambient equational theory;
- if \( u \) is of the form \( y(\ldots, z \leftarrow a(b \leftarrow r), \ldots) \), then
  \[
  u[b/a] = y(\ldots, z \leftarrow r, \ldots),
  \]
  as in equation (3.1.4) on the preceding page, and per equation (3.1.2) on the previous page, but we also add the equality \( b = z \) to the ambient equational theory;
- otherwise, if \( u \) is of the form \( y(z_1 \leftarrow v_i) \), and if the previous cases do not apply (i.e. \( a \) is not the front variable of \( u \) or \( v_i \) for all \( i \)), then
  \[
  u[b/a] = y(z_i \leftarrow v_i[b/a]),
  \]
  as in equation (3.1.4) on the preceding page.

From the formulation of system \( \text{OPT} \), it is clear that a sequent that is equivalent to a derivable one is itself derivable. Let us now turn our attention to rule \textbf{shift} above. It takes a term \( t \), thought of as a pasting diagram, and creates a new variable having \( t \) as source. One may thus think of it as a rule creating "fillers", akin to Kan filler condition on simplicial sets.

**Example 3.1.5.** Consider the term \( t = \alpha(g \leftarrow \beta) \) in a suitable context \( \Gamma \):

\[
\begin{align*}
  & f \quad \alpha \quad z \\
  & h \quad \beta \\
  & y \quad \frac{h}{x} \quad \frac{\alpha}{\beta} \quad z
\end{align*}
\]

Then the graftings \( t(f \leftarrow \gamma) = \alpha(f \leftarrow \gamma, g \leftarrow \beta) \) and \( t(i \leftarrow \gamma') = \alpha(g \leftarrow \beta(i \leftarrow \gamma)) \), for some appropriate \( \gamma \) and \( \gamma' \), can respectively be represented as:

\[
\begin{align*}
  & f \quad \gamma \quad \alpha \quad z \\
  & h \quad \beta \\
  & y \quad \frac{h}{x} \quad \frac{\alpha}{\beta} \quad z
\end{align*}
\]

**Example 3.1.6.** Consider the variables \( \alpha, \beta_x, \) and \( \beta_y \) typed as:

\[
\begin{align*}
  \alpha : (y \leftarrow f) & \mapsto x \mapsto \emptyset, & \beta_x : z \mapsto x \mapsto \emptyset, & \beta_y : y \mapsto y \mapsto \emptyset.
\end{align*}
\]

Then \( \alpha(f \leftarrow \beta_x) \) and \( \alpha(g \leftarrow \beta_y) \) can respectively be represented as:

\[
\begin{align*}
  & y \quad \frac{f}{o} \\
  & x \quad \frac{\alpha}{\beta} \\
  & z \quad \frac{h}{x} \quad \frac{\alpha}{\beta} \quad \beta
\end{align*}
\]

Then the sources \( \alpha(f \leftarrow \beta_x) \) and \( \alpha(g \leftarrow \beta_y) \) are respectively

\[
\begin{align*}
  g(y \leftarrow f)[z/f] & = g(y \leftarrow z), & g(y \leftarrow f)[y/g] & = y(y \leftarrow f) = f,
\end{align*}
\]

and, in the first case, the equation \( x = y \) is added to the ambient equational theory.

**Remark 3.1.7.** The \textbf{degen} rule may be replaced by the following \textbf{degen-shift} rule without changing the set of derivable sequents of the form \( (E \vdash \Gamma \vdash y : T) \) with \( y \in \mathbb{V} \): if \( x \in \mathbb{V}_n \) and \( d \in \mathbb{V}_{n+2} \) such that \( d \notin \mathbb{V}_{n+2} \), then

\[
\begin{align*}
  & E \vdash \Gamma \vdash x : T \\
  & E \vdash \Gamma, d : x \leadsto x \leadsto T \vdash \Gamma_{n+2} \vdash d : x \leadsto x \leadsto T \quad \text{degen-shift}
\end{align*}
\]

However, note that sequents of the form \( (E \vdash \Gamma \vdash y : T) \) are no longer derivable.

\footnote{The case in which \( u \) is of the form \( y(\ldots, z \leftarrow a(b_1 \leftarrow r_1, \ldots, b_k \leftarrow r_k), \ldots) \) with \( k > 1 \) does not happen in valid derivations.}
3.1.3. **Uniqueness of typing.** Let \((E \vdash \Gamma \vdash x : X)\) be a derivable sequent. We prove in theorem 3.1.8 that the type \(X = (sx \leftrightarrow ssx \leftrightarrow \cdots)\) is completely determined by \(sx\) and \(\Gamma\).

A consequence of this result is that at any stage, a context \(\Gamma\) may be replaced by its “meager form” \(\tilde{\Gamma}\), obtained by replacing “full typings” \(y : Y\) by \(y : sy\), i.e. by removing all but the top term of \(Y\). Using meager context comes with a cost however: checking the hypothesis of rule **graft** requires to compute the second source \(ssx\) of \(x\), which is not contained in \(\tilde{\Gamma}\). For clarity, we do not make use of meager forms throughout the rest of this work.

Define the function \(\tilde{s}\) as follows:

\[
\tilde{s} : \mathcal{T}_\Gamma \rightarrow \mathcal{T}_\Gamma
\]

\[
x \mapsto sx
\]

\[
x \mapsto x
\]

\[
x(y_i \leftarrow u_i') \mapsto (sx)[\tilde{s}u_i/i]
\]

where \(x, y_i \in \mathcal{V}_\Gamma, u_i' \in \mathcal{T}_\Gamma\).

**Theorem 3.1.8.** Let \((E \vdash \Gamma \vdash t : s_1 \leftrightarrow s_2 \leftrightarrow \cdots \leftrightarrow s_n \leftrightarrow \emptyset)\) be a derivable sequent. Then for \(1 \leq k \leq n\) we have \(\tilde{s}_k = \tilde{s}^k t\), or equivalently, for \(0 \leq i \leq n\), we have \(\tilde{s}_i = s_{i+1}\), with \(s_0 := t\).

**Proof.** We proceed by induction on the proof tree of the sequent. For readability, we omit equational theories and contexts.

1. If the sequent is obtained by the following proof tree:

\[
\vdash x : \emptyset
\]

then \(\tilde{s}x = sx = \emptyset\), since \(x \in \mathcal{V}\).

2. If the sequent is obtained by the following proof tree:

\[
s_1 : s_2 \leftrightarrow \cdots \leftrightarrow s_n \leftrightarrow \emptyset
\]

\[
t : s_1 \leftrightarrow s_2 \leftrightarrow \cdots \leftrightarrow s_n \leftrightarrow \emptyset\]

then \(s_1 \in \mathcal{V}\) and \(t = s_1\). Thus, \(\tilde{s}t = s_1\), while for \(1 \leq i \leq n\), the equality \(\tilde{s}_i = s_{i+1}\) holds by induction.

3. If the sequent is obtained by the following proof tree:

\[
s_1 : s_2 \leftrightarrow \cdots \leftrightarrow s_n \leftrightarrow \emptyset
\]

\[
t : s_1 \leftrightarrow s_2 \leftrightarrow \cdots \leftrightarrow s_n \leftrightarrow \emptyset\]

shift then \(t \in \mathcal{V}\), so \(\tilde{s}t = st = s_1\), while for \(1 \leq i \leq n\), the equality \(\tilde{s}_i = s_{i+1}\) holds by induction.

4. Assume now that the sequent is obtained by the following proof tree:

\[
u : r_1 \leftrightarrow r_2 \leftrightarrow \cdots \leftrightarrow r_n \leftrightarrow \emptyset \vdash x : X
\]

\[
t : s_1 \leftrightarrow s_2 \leftrightarrow \cdots \leftrightarrow s_n \leftrightarrow \emptyset\]

graft-\(a\) with \(a \in r_i^*\) and \(x \in \mathcal{V}\) such that \(ssx = sa\). Then \(t = u(a \leftarrow x)\), \(s_1 = r_1[ssx/a]\), and \(s_i = r_i\) for \(2 \leq i \leq n\). On the one hand, we have

\[
\tilde{s}_1 = \tilde{s}(u(a \leftarrow x))
\]

\[
= \tilde{s}[u[ssx/a]]
\]

\[
= \tilde{s}[ssx/a]
\]

since \(x \in \mathcal{V}\)

\[
= s_1.
\]

On the other hand, write \(r_1 = v(y \leftarrow a(z_i \leftarrow \overline{w}_i))\), for some \(v, \overline{w}_i \in \mathcal{T}_{n-1}\) and \(y \in \mathcal{V}_{n-2}\). Then

\[
\tilde{s}_1 = \tilde{s}(r_1[ssx/a])
\]

\[
= \tilde{s}\left(v(y \leftarrow (sx)(z_i \leftarrow \overline{w}_i))\right)
\]

\[
= \tilde{s}v[ssx[\tilde{s}u_i/i]/z_i/y]
\]

\[
= \tilde{s}v[ssx[\tilde{s}u_i/i]/z_i/y]
\]

by ind.

\[
= \tilde{s}v[ssx[\tilde{s}a]/z_i]/y]
\]

hyp. of graft-\(a\)

\[
= \tilde{s}r_1 = r_2 = s_2.
\]

Finally, for \(1 \leq i \leq n\), the equality \(\tilde{s}_i = s_{i+1}\) holds by induction. \(\square\)
Corollary 3.1.9. Let \((E ∷ Γ ⊢ t : T)\) be a derivable sequent, and \(x : s_1 ⇝ s_2 ⇝ ⋯ ⇝ s_n ⇝ ∅\) be a typing in \(Γ\). Then for \(1 ≤ k ≤ n\) we have \(s_k = t_k\), or equivalently, for \(0 ≤ i ≤ n\), we have \(s_{i+1} = i\), with \(s_0 := x\).

Proof. If \(x : s_1 ⇝ s_2 ⇝ ⋯ ⇝ s_n ⇝ ∅\) is a typing in \(Γ\), then somewhere in the proof tree of \((E ∷ Γ ⊢ t : T)\) appears a sequent of the form \((F ∷ T ⊢ x : s_1 ⇝ s_2 ⇝ ⋯ ⇝ s_n ⇝ ∅)\), which is necessarily derivable. We conclude by applying theorem 3.1.8 on the preceding page.

By definition, \(s\) extends \(s\) to a function \(T_Γ ⊢ T_Γ\), and for convenience, we just write it as \(s\) in the sequel, and call it the source of a term.

Example 3.1.10. Consider the following term:

\[
α(f ← γ, g ← β) = \begin{array}{c}
α
\end{array}
\]

Then its source is computed as follows:

\[
s(α(f ← γ, g ← β)) = (sα)(sγ) / f, (sβ) / g
\]

Since \(sα = g(y ← f)
\]

\[
= (g(y ← f))[(sγ) / f, (sβ) / g]
\]

Since \(sγ = j
\]

\[
= (g(y ← f))[(j / f, (sβ) / g)]
\]

Since \(sβ = i(z ← h)
\]

\[
= (g(y ← j))[i(z ← h) / g]
\]

\[
= (i(z ← h))(y ← j)
\]

\[
= i(z ← h)(y ← j)
\]

Since \(y ∈ (sh)^∗\).

The latter term indeed corresponds to the source of the pasting diagram, i.e. the arrow composition on the top.

3.2. Equivalence with polynomial opetopes

In this section, all sequents are assumed derivable in \(OPT^1\). We show that sequents typing a variable (up to \(\sim\)) are in bijective correspondence with polynomial opetopes (see section 2.2 on page 10). To this end, we define the polynomial coding operation \(\llbracket - \rrbracket_{n+1}^{poly}\) that maps a sequent \((E ∷ Γ ⊢ t : T)\), with \(t ∈ T_n\), to an \((n + 1)\)-opetope \(\llbracket E ∷ Γ ⊢ t : T \rrbracket_{n+1}^{poly}\) or even \(\llbracket t \rrbracket_{n+1}^{poly}\) for short, if no ambiguity arises. Also, if \(α ∈ V_{n+1}\), then we set \(\llbracket α \rrbracket_{n+1}^{poly} := [sα]_{n+1}^{poly}\).

The idea of the polynomial coding is to map a pasting diagram described by a term (on the left) to its underlying composition tree, and reapply the coding recursively (on the right):

\[
\llbracket α(g ← β) \rrbracket = \begin{array}{c}
\llbracket α \rrbracket
\end{array}
\]

where \(\lfloor \cdot \rfloor\) is a shorthand for \(\llbracket \cdot \rrbracket_{poly}\).

For \(t = x(y_i ← u_i) ∈ T_n\) and \(z ∈ t^∗\), the address \(ιz ∈ \mathbb{A}_n\) of \(z\) in \(t\) is an \(n\)-address (see section 2.2.3 on page 11) that indicates “where \(z\) is located in \(t\).

Definition 3.2.1 (Address). Take \(t ∈ T_Γ\), \(t = x(y_i ← u_i)\).

1. For \(z ∈ t^∗\), the address \(ιz ∈ \mathbb{A}_n\) of \(z\) in \(t\) is given by

\[
ιz := \begin{cases}
\emptyset & \text{if } z = x \text{ in the current eq. th.,}
\mathbb{A}_n \cdot y_i, \cdot u_i, \cdot z & \text{if } z ∈ u_i^∗.
\end{cases}
\]

If \([p] = ιz\), then we write \(s_{[p]} := z\). In particular, \(s_I t = x\).

2. For \(a ∈ (st)^∗\), the address \(ιa ∈ \mathbb{A}_n\) of \(a\) in \(t\) is given by

\[
ιa := \begin{cases}
\mathbb{A}_n \cdot a & \text{if } a ∈ (sx)^∗,
\mathbb{A}_n \cdot y_i, \cdot u_i, \cdot a & \text{if } a ∈ (sx)^∗, \text{ but } a ∈ (su_i)^∗.
\end{cases}
\]
Example 3.2.2. The context describing the following pasting scheme:

contains the following typings: \( x, y, z : \emptyset, f : x \rightarrow \emptyset, g : y \rightarrow \emptyset, h : y \rightarrow \emptyset, i : z \rightarrow \emptyset, \alpha : g(y \leftarrow f) \rightarrow a \rightarrow \emptyset, \) and \( \beta : i(z \leftarrow h) \rightarrow b \rightarrow \emptyset. \) Then for \( t := \alpha(g \leftarrow \beta), \) we have

\[
\begin{align*}
&\&_{t}\alpha = [], \\
&\&_{t}\beta = [\&_{s}g], \&_{t}\beta = [[[]] \cdot [] = [[]]], \\
&\&_{t}i = [\&_{s}g], \&_{t}i = [[[]] \cdot [[]] = [[]] \cdot [[]] = [[]]]. \\
&\&_{t}h = [\&_{s}g], \&_{t}h = [[[]] \cdot [\&_{s}h] = [[[]] \cdot [\&_{s}h] \cdot \&_{h}h = [[[[]]]] = [[]]]; \\
&\&_{t}f = [\&_{s}g, \&_{t}f] = [[[]]].
\end{align*}
\]

Those addresses indeed match with that of the corresponding opetope:

Definition 3.2.3 (Polynomial coding). The polynomial coding operation \( [\cdot]_{n}^{\text{poly}} \) is defined inductively by:

\[
\begin{align*}
[x : \emptyset]_{0}^{\text{poly}} &:= \diamond, & x &\in \mathcal{V}_{0}, \quad (3.2.4) \\
[x : x \rightarrow \cdots]_{n+2}^{\text{poly}} &:= [x]_{n}^{\text{poly}}, & x &\in \mathcal{V}_{n}, \quad (3.2.5) \\
[x(y_{1} \leftarrow u_{1}) : \cdots]_{n+1}^{\text{poly}} &:= \mathcal{Y}_{x}^{\text{poly}} \circ [u_{1}]_{n+1}^{\text{poly}}, & x(y_{1} \leftarrow u_{1}) &\in \mathcal{T}_{n}. \quad (3.2.6)
\end{align*}
\]

We recall that (see equation (2.1.6) on page 10)

\[
\mathcal{Y}_{x}^{\text{poly}} \circ [u_{1}]_{n+1}^{\text{poly}} := \left(\cdots \mathcal{Y}_{x}^{\text{poly}} \circ [\&_{s}y_{1}] \circ [u_{1}]_{n+1}^{\text{poly}} \circ [\&_{s}y_{2}] \circ [u_{2}]_{n+1}^{\text{poly}} \cdots \circ [\&_{s}y_{n}] \circ [u_{n}]_{n+1}^{\text{poly}}\right).
\]

It is clear that the coding function is well defined in equations (3.2.4) and (3.2.5). We now establish a series of results to prove proposition 3.2.12 on page 23 stating that equation (3.2.6) is well defined too.

Proposition 3.2.7. For \( t \in \mathcal{T}_{x_{1}}^{n}, \) \( t = x(\overline{y_{1}} \leftarrow \overline{u_{1}}), z \in t^{*}, \) and \( [p] := \&_{t}z, \) we have \( s_{[p]}[t]_{n+1}^{\text{poly}} = [z]_{n}^{\text{poly}}. \)

Proof. By definition we have

\[
[t]_{n+1}^{\text{poly}} = \mathcal{Y}_{[x]_{n}^{\text{poly}}} \circ [u_{1}]_{n+1}^{\text{poly}},
\]

and we distinguish two cases. If \( z = x, \) then \( [p] = [], \) and the result clearly holds. Otherwise, \( [p] = [\&_{s}x y_{j}] \cdot \&_{u_{j}}z, \)

for \( j \) such that \( z \in u_{j}^{*}. \) Then,

\[
\begin{align*}
s_{[p]}[t]_{n+1}^{\text{poly}} &= s_{[\&_{s}x y_{j}] \cdot \&_{u_{j}}z} \left(\mathcal{Y}_{[x]_{n}^{\text{poly}}} \circ [u_{1}]_{n+1}^{\text{poly}}\right) \\
&= s_{\&_{u_{j}}z} [u_{j}]_{n+1}^{\text{poly}} \\
&= [z]_{n}^{\text{poly}}
\end{align*}
\]

by induction.

\[\square\]

Corollary 3.2.8. For \( x \) as in case (3.2.6) above, and \( [q_{i}] := \&_{s}x y_{i}, \) we have \( s_{[q_{i}]}[x]_{n}^{\text{poly}} = [y_{i}]_{n-1}^{\text{poly}}. \)
Proof. We have \( s_{[y]} x \_n^{\text{poly}} = s_{[y]} x \_n^{\text{poly}} = y_1 \_n^{\text{poly}} \). \( \square \)

**Lemma 3.2.9.** Consider the following derivation tree:

\[
\vdots \vdash_n t : T \quad \vdots \vdash_n x : X \quad \text{graft-}a
\]

Writing \( r := t(a \leftarrow x) \), we have \( [r]_n^{\text{poly}} = [t]_n^{\text{poly}} \circ_{\&_t} a Y_\_x \_n^{\text{poly}} \).

**Proof.** We have \( [r]_n^{\text{poly}} = [t]_n^{\text{poly}} \circ_{\&_t} a Y_\_x \_n^{\text{poly}} \), for some address \([l]\). Then, writing \( t = z(y_i \leftarrow u_i) \), we have, according to definition 3.1.1 on page 17,

\[
r = z(y_i \leftarrow u_i)(a \leftarrow x) = \begin{cases} z(y_i \leftarrow u_i, a \leftarrow x) & \text{if } a \in (s \_z)^*, \\ z(y_i \leftarrow u_i(a \leftarrow x)) & \text{if } a \notin (s \_z)^*. \end{cases}
\]

There are two cases:

1. if \( a \in (s \_z)^* \), then \([l] = [\&_z a] = \&_t a\);
2. if \( a \notin (s \_z)^* \), then

\[
[r]_n^{\text{poly}} = Y_\_z \_n^{\text{poly}} \circ_{\&_z y_i} [u_i(a \leftarrow x)]_n^{\text{poly}}.
\]

Let \( j \) be the index such that \( a \in (s u_j)^* \). Then, by induction,

\[
[u_j(a \leftarrow x)]_n^{\text{poly}} = [u_j]_n^{\text{poly}} \circ_{\&_{u_j} a} Y_\_x \_n^{\text{poly}},
\]

thus \([l] = [\&_z y_j] \cdot \&_{u_j} a = \&_t a\). \( \square \)

**Lemma 3.2.10** (Named readdressing lemma). Let \( r \in T_{\_1, \_n} \). Then for \( a \in (s \_r)^* \), we have \( \&_s b = \varphi_{\_1} \_n^{\text{poly}} \&_r b \), where \( \varphi \) is the readdressing function introduced in section 2.1.1 on page 10.

**Proof.** Recall the definition of the readdressing map from appendix A on page 71. If \( r \) is a variable, then

\[
\&_s b = \varphi_{\_1} \_n^{\text{poly}} [\&_s b] = \varphi_{\_1} \_n^{\text{poly}} \&_r b.
\]

Otherwise, write the derivation tree of \( r \):

\[
\vdots \vdash_n t : T \quad \vdots \vdash_n x : X \quad \text{graft-}a
\]

assume that \( \&_s a = \varphi_{\_1} \_n^{\text{poly}} \&_t a \) holds by induction. Since \( s \_r = s \_t [s \_x / a] \), we have

\[
\&_s b = \begin{cases} \&_t a \cdot \&_s b & \text{if } b \in (s \_x)^*, \\ \&_t a \cdot \&_s c \cdot [p] & \text{if } b \in (s \_t)^*, \&_t a \subseteq \&_s b, \\ \varphi_{\_1} \_n^{\text{poly}} & \text{say } \&_s b = \varphi_{\_1} \_n^{\text{poly}} \&_t a \cdot \[\&_s a c \cdot [p], \\ \&_s b & \text{if } b \in (s \_t)^*, \&_s a \neq \&_t b. \end{cases}
\]

\( \varphi_{\_1} \_n^{\text{poly}} \&_t a \cdot \&_s b = \varphi_{\_1} \_n^{\text{poly}} \&_r b. \) \( \square \)

**Proposition 3.2.11.** Let \( a \in V_n \) be a typed \( n \)-variable, for \( n \geq 2 \). Then \( t[a]_n^{\text{poly}} = [ss a]_n^{\text{poly}} \).

**Proof.** We proceed by induction on \( a \).

1. If \( a = x \in V_{n-1} \), then \( [a]_n^{\text{poly}} = Y_\_x \_n^{\text{poly}} \), and \( t[a]_n^{\text{poly}} = [x]_n^{\text{poly}} = [s x]_n^{\text{poly}} = [ss a]_n^{\text{poly}} \).
2. If \( a = a \) for some \( a \in V_{n-2} \), then \( [a]_n^{\text{poly}} = [a]_n^{\text{poly}} \), and \( t[a]_n^{\text{poly}} = Y_\_a \_n^{\text{poly}} = [a]_n^{\text{poly}} = [ss a]_n^{\text{poly}} \).
3. Otherwise, \( a \) is given by the following proof tree:

\[
\vdots \vdash_n t : T \quad \vdots \vdash_n x : X \quad \text{graft-}a
\]
and write \( r := s \alpha = t(a \rightsquigarrow x) \) for shorter notations. By lemma 3.2.9 on the facing page we have
\[
[r]_{n+1}^{\text{poly}} = [t]_{n+1}^{\text{poly}} \circ \kappa_{t,a} [s]_{n}^{\text{poly}},
\]
and finally, for \( \circ \) the partial multiplication of \( \mathcal{O}^n \) as in theorem A.1.9 on page 72,
\[
[s\alpha]_{n}^{\text{poly}} = [s]_{n}^{\text{poly}} \circ [\alpha]_{n+1}^{\text{poly}} \quad \text{by definition}
\]
\[
= [st]_{n}^{\text{poly}} \circ_{\kappa_{t,a}} [s x]_{n}^{\text{poly}} \quad \text{by induction}
\]
\[
= t[tt]_{n+1}^{\text{poly}} \circ_{\kappa_{t,a}} [x]_{n}^{\text{poly}} \quad \text{with } [p] = p[t]_{n+1}^{\text{poly}} (\&_{t,a})
\]
\[
= t\left( t[tt]_{n+1}^{\text{poly}} \circ_{\kappa_{t,a}} \mathcal{Y}_{e_{n}}^{\text{poly}} \right) \quad \text{by lemma 3.2.10}
\]
\[
= t[\alpha]_{n+1}^{\text{poly}}.
\]
\(\Box\)

**Proposition 3.2.12.** With variables as in equation (3.2.6) on page 21, we have that for all \( i \)
\[
[ts]_{i} [u_{i}]_{n+1}^{\text{poly}} = s_{\kappa_{t,y_{i}}} [x]_{n}^{\text{poly}},
\]
and the graftings are well defined.

**Proof.** Write \( u_{i} := a(\overrightarrow{b_{i}} \rightsquigarrow \overrightarrow{v_{i}}) \). Then
\[
[ts]_{i} [u_{i}]_{n+1}^{\text{poly}} = [a]_{n}^{\text{poly}} \quad \text{by proposition 3.2.7}
\]
\[
= [s\alpha]_{n-1}^{\text{poly}} \quad \text{by proposition 3.2.11}
\]
\[
= [s y_{i}]_{n-1}^{\text{poly}} \quad \text{by graft rule}
\]
\[
= [y_{i}]_{n-1}^{\text{poly}} \quad \text{by definition}
\]
\[
= s_{\kappa_{t,y_{i}}} [x]_{n}^{\text{poly}} \quad \text{by corollary 3.2.8}.
\]
\(\Box\)

This result concludes the proof that equations (3.2.4) to (3.2.6) of definition 3.2.3 on page 21 are well defined.

**Corollary 3.2.13.** Let \((E \vdash \Gamma \vdash n t : T)\) be a derivable sequent. Then \(\&_{t}\) exhibits a bijection between the set of \(n\)-variables of \(t\), and \(\left( [t]_{n+1}^{\text{poly}} \right)^{*}\). It is also a bijection between the set of \((n-1)\)-variables of \(st\) and \(\left( [t]_{n+1}^{\text{poly}} \right)^{1}\).

The rest of this section is dedicated to prove theorem 3.2.22 on page 25 stating that \( [\cdot]_{n}^{\text{poly}} \) is a bijection modulo \(\varepsilon\). We first prove surjectivity, by defining a sequent \( [\omega]_{n}^{\text{L}} \) such that \( [\omega]_{n}^{\text{poly}} = \omega \), for any opetopes \(\omega \in \mathcal{O}_{n}\). We proceed by opetopic induction (see remark 2.2.3 on page 12).

1. Trivially, \( [\bullet]_{n}^{\text{L}} \) is obtained by the following proof tree:
\[
\begin{array}{c}
\hline
[\bullet]_{n}^{\text{L}} \\
\hline
\text{point}
\end{array}
\]  
(3.2.14)

with an arbitrary choice of variable (different choices lead to equivalent sequents).

2. For \( \phi \in \mathcal{O}_{n-2} \) the sequent \( [L_{\phi}]_{n}^{\text{L}} \) is obtained by the following proof tree:
\[
\begin{array}{c}
\hline
[L_{\phi}]_{n}^{\text{L}} \\
\hline
\text{degen}
\end{array}
\]  
(3.2.15)

3. For \( \psi \in \mathcal{O}_{n-1} \), the sequent \( [Y_{\psi}]_{n}^{\text{L}} \) is obtained by the following proof tree:
\[
\begin{array}{c}
\hline
[Y_{\psi}]_{n}^{\text{L}} \\
\hline
\text{shift}
\end{array}
\]  
(3.2.16)
with an arbitrary choice of fresh variable (different choices lead to equivalent sequents).

(4) Let \( \nu \in \text{Dom}\), having at least one node, \([l] \in \nu\), and \( \psi \in \text{Dom}_{n-1}\) be such that the grafting \( \nu \circ [l] \triangleright \psi \) is well-defined. Then the sequent \( \nu \circ [l] \triangleright \psi \) is obtained by the following proof tree:

\[
\frac{[\nu]}{[\nu \circ [l] \triangleright \psi]} \text{ graft-\( \alpha \)} \tag{3.2.17}
\]

where \( [\nu] = (E \triangleright \Gamma \vdash u : U) \), where the variable \( \alpha \in (ss w)^{*} \) is such that \( \&_{ss w} \alpha = [l] \) (see corollary 3.2.13 on the preceding page), and where the adequate \( \alpha \)-conversion have been performed to fulfill the side conditions of graft. We check in propositions 3.2.19 and 3.2.20 that this definition is well-founded.

**Lemma 3.2.18.** Let \( (E \triangleright \Gamma \vdash w : V) \vdash [(\omega')] \) with \( \omega \in \text{Dom}_n \) non degenerate, \( n \geq 2 \). Then, by corollary 3.2.13 on the preceding page, for \([l] \in \omega\), there is a variable \( \alpha \in (ss w)^{*} \) such that \( \&_{ss w} \alpha = [l] \).

**Proof.** By assumption, \( \omega \) is either of the form \( \triangleright \psi \), or \( \nu \circ [k] \triangleright \psi \), for some \( \nu \in \text{Dom}_n \) and \( \psi \in \text{Dom}_{n-1} \). From there, this is a straightforward induction.

**Proposition 3.2.19.** In proof tree (3.2.6), the instance of graft is well-defined.

**Proof.** Write \( [\psi] = (F \triangleright Y \vdash \text{Dom}_{n-1} \triangleright p : P) \). By the previous lemma, we have \( \alpha \in (ss w)^{*} \). Further,

\[
\begin{align*}
[sa]_{n-2} \text{poly} & = [e[l] u]_{n-2} \text{poly} \\
& = [e[l] u]_{n} \text{poly} & \text{ by proposition 3.2.7 on page 21} \\
& = [e[l] u]_{n} & \text{ by relation (Inner), see section 2.2.4} \\
& = t [s[l] \triangleright \psi] & \text{ by corollary 3.2.7 on page 21} \\
& = [ss p]_{n-2} \text{poly} & \text{ by proposition 3.2.11 on page 22.}
\end{align*}
\]

By induction on \( n \), the polynomial coding \( [\cdot]_{n-2} \text{poly} \) is injective modulo \( \equiv \). Hence, we can assume \( sa = ss p \) without loss of generality, and finally, the instance of the graft rule is well-defined.

**Proposition 3.2.20.** Let \( \omega \in \text{Dom}_n \) be a non degenerate opetope, and consider two arbitrary decompositions in corollas

\[
\omega = \left( \cdots \left( y_{s[p_1]}^{\omega} \circ_{\{p_2\}} y_{s[p_2]}^{\omega} \right) \circ_{\{p_3\}} y_{s[p_3]}^{\omega} \cdots \right) \circ_{\{p_k\}} y_{s[p_k]}^{\omega}
\]

Then
\[
\left[ \left( y_{s[p_1]}^{\omega} \circ_{\{p_2\}} y_{s[p_2]}^{\omega} \right) \cdots \circ_{\{p_k\}} y_{s[p_k]}^{\omega} \right] \]

In other words, \( [\omega] \) does not depend on the decomposition of \( \omega \) in corollas.

**Proof.** By assumption, the sequence \([p_1], \ldots, [p_k]\) (and likewise for \([q_1], \ldots, [q_k]\)) has the following property: for \( 1 \leq i \leq j \leq k \), either \([p_i] \leq [p_j]\) or \([p_j] \leq [p_i]\) are \( \leq \)-incomparable (recall that \( \leq \) is the lexicographical order on \( \text{Dom}_{n-1} \), see section 2.2.3 on page 11). Further, \([p_1], \ldots, [p_k]\) = \( \omega^{*} = ([q_1], \ldots, [q_k]) \). Consequently, the sequence of addresses \([q_1], \ldots, [q_k]\) can be obtained from \([p_1], \ldots, [p_k]\) via a sequence of transpositions of consecutive \( \leq \)-incomparable addresses.

It is thus enough to check the following: for \( \nu \in \text{Dom}_n \), \([l], [\nu] \in \nu\) (necessarily, neither is a prefix of the other), and \( \psi, \psi' \in \text{Dom}_{n-1} \) such that the following graftings are well defined, we have

\[
\left[ \left( \nu \circ [l] \triangleright \psi \right) \circ_{[\nu]} \psi' \right] = \left[ \left( \nu \circ_{[\nu]} \psi' \right) \circ [l] \triangleright \psi \right].
\]
Let $[[\nu]] = (E \triangleright \Gamma \vdash x_\nu : s_\nu \leftrightarrow X_\nu)$, and likewise for $\psi$ and $\psi'$, and $a, a' \in (55\nu)^\ast$ be such that $\&_{a+s}a = [t]$ and $\&_{a+s}a' = [t']$ (see corollary 3.2.13 on page 23). The sequents above are respectively obtained by the following proof trees:

$$\begin{align*}
E_\nu \triangleright \Gamma_\nu \vdash x_\nu : s_\nu \leftrightarrow X_\nu & \quad E_\psi \triangleright \Gamma_\psi \vdash x_\psi : s_\psi \leftrightarrow X_\psi \\
F \triangleright \Gamma_\nu \vdash \Gamma_\nu \vdash \Gamma_\psi \vdash x_\nu(a \leftarrow x_\psi) : s_\nu[s_\nu/a] \leftrightarrow X_\nu & \quad G \triangleright \Gamma_\nu \vdash \Gamma_\nu \vdash \Gamma_\psi \vdash x_\nu(a' \leftarrow x_\psi) : s_\nu[s_\nu/a'][s_\nu/a'] \leftrightarrow X_\nu
\end{align*}$$

$$\begin{align*}
E_\nu \triangleright \Gamma_\nu \vdash x_\nu : s_\nu \leftrightarrow X_\nu & \quad E_\psi \triangleright \Gamma_\psi \vdash x_\psi : s_\psi \leftrightarrow X_\psi \\
F' \triangleright \Gamma_\nu \vdash \Gamma_\nu \vdash \Gamma_\psi \vdash x_\nu(a' \leftarrow x_\psi) : s_\nu[s_\nu/a'][s_\nu/a'] \leftrightarrow X_\nu & \quad G' \triangleright \Gamma_\nu \vdash \Gamma_\nu \vdash \Gamma_\psi \vdash x_\nu(a' \leftarrow x_\psi)(a \leftarrow x_\psi) : s_\nu[s_\nu/a'][s_\nu/a] \leftrightarrow X_\nu
\end{align*}$$

It remains to prove that both these conclusions sequents are equal.

1. Since by assumptions $[t], [t'] \in \nu^\ast$, we have $a \notin s_\nu^\ast$ and $a' \notin s_\nu^\ast$. Thus

$$x_\nu(a \leftarrow x_\psi)(a' \leftarrow x_\psi) = x_\nu(a \leftarrow x_\psi, a' \leftarrow x_\psi) = x_\nu(a' \leftarrow x_\psi)(a \leftarrow x_\psi).$$

2. Again, since $a \notin s_\nu^\ast$ and $a' \notin s_\nu^\ast$, we have

$$s_\nu[s_\nu/a][s_\nu/a'] = s_\nu[s_\nu/a][s_\nu/a].$$

3. Lastly, the equational theories $G$ and $G'$ are the union of $E_\nu$, $E_\psi$, and $E_\psi'$, and the potential additional equalities incurred by the independent substitutions $s_\nu/a$ and $s_\nu/a'$. Hence $G = G'$.

**Corollary 3.2.21.** For any opetope $\omega \in \Omega$, the sequent $[[\omega]]$ is uniquely defined up to $\sim$.

**Proof.** Clearly, proof trees (3.2.14), (3.2.15), and (3.2.16) are well-defined. In proposition 3.2.19 on the preceding page, we show that the same holds for proof tree (3.2.17). Finally, in proposition 3.2.20 on the facing page, we show that for a non degenerate opetope $\omega \in \Omega_\nu$, the sequent $[[\omega]]$ does not depend on the decomposition of $\omega$.

**Theorem 3.2.22.** The polynomial coding $[[\cdot]]_n^{poly}$ is a bijection modulo $\sim$, whose inverse is $[[\cdot]]_n^{poly}$ restricted to $\Omega_n$.

**Proof.** We first show that for $\omega \in \Omega_n$, we have $[[\omega]]_n^{poly} = \omega$ by opetopic induction (see remark 2.2.3 on page 12).

1. By definition of $[[\cdot]]_n^{poly}$, $[[\nu]]_0^{poly} = \nu$.
2. With the same notations as in (3.2.15), and by induction, we have

$$[[l_\phi]]_n^{poly} = [[l_\phi]]_n^{poly} = l_\phi.$$

3. With the same notations as in (3.2.16), and by induction, we have

$$[[Y_\psi]]_n^{poly} = Y_\psi[[v]]_{n-1}^{poly} = Y_\psi.$$

4. With the same notations as in (3.2.17), and by induction, we have

$$[[\nu \circ Y_\psi]]_n^{poly} = [[\nu]]_n^{poly} \circ [[Y_\psi]]_n^{poly} = [[\nu]]_n^{poly} \circ [[Y_\psi]]_n^{poly} = \nu \circ Y_\psi.$$ 

Conversely, we now show that for all sequents $(E \triangleright \Gamma \vdash a : T)$ (abbreviated $(\alpha : T)$ if no ambiguity arise) we have $(E \triangleright \Gamma \vdash \alpha : T) = [[E \triangleright \Gamma \vdash \alpha : T]]_n^{poly}$ up to $\sim$.

1. We have that $[[x : \varnothing]]_0^{poly} = [\varnothing] \simeq (x : \varnothing)$.
2. With the same notations as in equation (3.2.5) on page 21, $[[\delta : \cdot \leftrightarrow x \leftrightarrow X]]_n^{poly} = [[l_{\varnothing \cdot X n}^{poly}]]$, and both sequents $[[l_{\varnothing \cdot X n}^{poly}]]$ and $(\delta : \cdot \leftrightarrow x \leftrightarrow X)$ are obtained by applying $\text{degen}$ to $(x : X)$. Thus $[[\delta : \cdot \leftrightarrow x \leftrightarrow X]]_n^{poly} \simeq (\delta : \cdot \leftrightarrow x \leftrightarrow X)$. 

Lastly, consider the sequent \( \rho(x(y_i \leftarrow u_i)) \rightarrow T \) as in equation (3.2.6) on page 21. Then
\[
\left[ \left[ \rho(x(y_i \leftarrow u_i)) \rightarrow T \right]_{\text{poly}} \right]^n_{n+1} = \left[ \bigcup_{k \in \mathbb{N}} \left[ u_i \right]_{n+1} \right]_{n+1} \rho(x(y_i \leftarrow u_i)) \rightarrow T'.
\]
Since \( T \) and \( T' \) are completely determined by \( x(y_i \leftarrow u_i) \) (see theorem 3.1.8 on page 19), we have that \( T = T' \), whence
\[
\left[ \left[ \rho(x(y_i \leftarrow u_i)) \rightarrow T \right]_{n+1} \right]_{n+1} \rho(x(y_i \leftarrow u_i)) \rightarrow T.
\]
\[\square\]

### 3.3. Examples

In this section, we showcase the derivation of some low dimensional opetopes. On a scale of a proof tree, specifying the context at every step is redundant. Hence we allow omitting it, only having the equational theory on the left of \( \vdash \).

**Example 3.3.1** (The arrow). The unique 1-opetope, the *arrow*, is given by the following simple derivation:

\[
\begin{array}{c}
\vdash 0 a: \emptyset \\
\vdash 1 f: a \rightarrow \emptyset \\
\vdash 1 a: a \leftarrow \emptyset
\end{array}
\]

**Example 3.3.2** (Opetopic integers). The set of \( \mathcal{O}_2 \) of 2-opetopes is in bijection with the set of natural numbers. Given \( n \in \mathbb{N} \), we denote by \( n \) the 2-opetope whose pasting diagram is a sequence of \( n \) arrows as follows:

\[
\begin{array}{c}
\vdash 0 a: \emptyset \\
\vdash 1 g: a \rightarrow \emptyset \\
\vdash 1 a: a \leftarrow \emptyset
\end{array}
\]

The derivation of the opetope \( 0 \) is

\[
\begin{array}{c}
\vdash 1 f_1: a_1 \leftarrow \emptyset \\
\vdash 1 f_2: a_2 \leftarrow \emptyset \\
\vdash 1 g: a_1 \rightarrow \emptyset
\end{array}
\]

alternatively, we could have used the **degen-shift** rule. For \( n \geq 1 \), the opetope \( n \) is derived as

\[
\begin{array}{c}
\vdash 1 f_1: a_1 \leftarrow \emptyset \\
\vdash 1 f_2: a_2 \leftarrow \emptyset \\
\vdash 1 f_3: a_3 \leftarrow \emptyset \\
\vdash 1 g: a_1 \rightarrow \emptyset
\end{array}
\]

**Example 3.3.3** (A classic). The 3-opetope

\[
\begin{array}{c}
\vdash 1 f_1: a_1 \leftarrow \emptyset \\
\vdash 1 f_2: a_2 \leftarrow \emptyset \\
\vdash 1 f_3: a_3 \leftarrow \emptyset \\
\vdash 1 g: a_1 \rightarrow \emptyset
\end{array}
\]

is derived as follows.

\[
\begin{array}{c}
\vdash 0 c: \emptyset \\
\vdash 1 h: c \leftarrow \emptyset \\
\vdash 1 i: a \leftarrow \emptyset
\end{array}
\]

\[
\begin{array}{c}
\vdash 1 h(c \leftarrow i): c[a/c] \leftarrow \emptyset \\
\vdash 1 a: \emptyset \\
\vdash 1 a: \emptyset
\end{array}
\]

and \( c[a/c] = a \). Then,
On the other hand we have

\[
\begin{align*}
\vdash h(c \leftarrow i) & : a \rightsquigarrow \varnothing \\
\vdash \beta: h(c \leftarrow i) & : a \rightsquigarrow \varnothing
\end{align*}
\]

The last grafting is well defined as \( s i = a = s s \alpha \), and \( h(c \leftarrow i)[g(b \leftarrow f)/i] = h(c \leftarrow g(b \leftarrow f)) \). Finally

\[
\begin{align*}
\vdash \beta(i \leftarrow \alpha): h(c \leftarrow g(b \leftarrow f)) & : a \rightsquigarrow \varnothing \\
\vdash A: \beta(i \leftarrow \alpha) & : h(c \leftarrow g(b \leftarrow f)) : a \rightsquigarrow \varnothing
\end{align*}
\]

Finally, the complete proof tree is:

Example 3.3.4 (A degenerate case). The 3-opetope on the left is derived as on the right:

Example 3.3.5 (Another degenerate case). The 3-opetope

is derived as follows:

\[ a = b + \beta(f \leftarrow \alpha): g(b \leftarrow f)[a/f] : a \rightsquigarrow \varnothing \]

and \( g(b \leftarrow f)[a/f] = g \), with the added equality \( a = b \).
3. NAMED APPROACH

\[
\begin{align*}
\frac{a = b \vdash \beta(f \leftarrow a) \colon g \dashv a \dashv \emptyset}{a = b \vdash A \colon \beta(f \leftarrow a) \dashv g \dashv a \dashv \emptyset}
\end{align*}
\]

\[\text{shift}\]

3.4. PYTHON IMPLEMENTATION

In this section, we briefly discuss the Python implementation [Ho Thanh, 2018b] of the present work. System OPT\(^1\) and all required syntactic constructs are implemented in module \texttt{opetopy.NamedOpetope}. The rules are represented by functions \texttt{point}, \texttt{degen}, \texttt{shift}, \texttt{graft}, as well as \texttt{degenshift} for the alternative rule presented in remark 3.1.7 on page 18. Those rules are further encapsulated in rule instance classes \texttt{Point}, \texttt{Degen}, \texttt{shift}, \texttt{Graft}, and \texttt{DegenShift}, which represent rule instances in a proof tree, so constructing a derivation amounts to writing a Python term using those four classes. If that term evaluates without raising any exception, then the proof tree is considered correct.

**Figure 3.4.1.** Derivation of the arrow sequent using \texttt{opetopy.NamedOpetope}

```python
from opetopy.NamedOpetope *

# We first derive the point that will act as the source of the arrow by invoking the point
# rule on variable "a".

a = Point("a")

# We then apply the shift rule on a by providing a fresh variable, here "f".

f = Shift(a, "f")

# Since we use names, the following sequent, while corresponding to the same opetope, is
# different from f

g = Shift(Point("b"), "g")

# Note that the function opetopy.NamedOpetope.Arrow can be used to concisely get a proof
# tree of ◾.

```

**Figure 3.4.2.** Derivation of some opetopic integers using \texttt{opetopy.NamedOpetope}, continuation of figure 3.4.1

```python
opetopic_integer_0 = DegenShift(a, "n_0")
opetopic_integer_1 = Shift(f, "n_1")
opetopic_integer_2 = Shift(Graft(g, f, "b"), "n_2")

# Note that the function opetopy.NamedOpetope.OpetopicInteger can be used to get the proof
# tree of an arbitrary opetopic integer.
```
3.5. THE SYSTEM FOR OPETOIC SETS

We now present OptSet, a derivation system for opetopic sets that is based on Opt. We first present the required syntactic constructs and conventions in section 3.1.1 on page 15, and present the inference rules in figure 3.5.1 on the next page.

3.5.1. Syntax. The main distinguishing feature of the named approach above is that only source faces of whichever cell is currently being derived are specified:

\[ \vdash n : s \Uparrow ss s \Downarrow \cdots \]

Nonetheless, as proven in proposition 3.2.11 on page 22, all the information about targets remain. To adapt our previous derivation system to opetopic sets, all faces, including targets, need to be explicitly specified. This will be part of the role of rule repr of system OptSet. Further, recall that a sequent in system Opt:

\[ E \triangleright \Gamma \vdash t : T. \]

An opetopic set is a set of opetopic cells, and in a sequent, the context and the equational theory suffice to describe this. Thus, system OptSet will only deal with expressions of the form \( E \triangleright \Gamma \) called opetopic contexts modulo theory (or OCMTs for short).

For example, the OCMT describing the following opetopic set (note that the graphical representation is not unique):

\[
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
 b \vdash f, a \vdash g = tt \alpha = th = tt \beta \\
 h \vdash t \beta = t \alpha
\end{array}
\end{array}
\end{array}
\end{array}
\]

is given by

\[
\begin{pmatrix}
 a : \emptyset, b : \emptyset, tf : \emptyset, tg : \emptyset, tt \alpha : \emptyset, th : \emptyset, tt \beta : \emptyset \\
 f : a \dashv \emptyset, g : a \dashv \emptyset, t \alpha : a \dashv \emptyset, h : a \dashv \emptyset, t \beta : a \dashv \emptyset \\
 \alpha : g(b \dashv f) \dashv a \dashv \emptyset, \beta : h \dashv a \dashv \emptyset
\end{pmatrix}
\]
3.5.2. **Inference rules.** Our derivation system for opetopic sets, presented in figure 3.5.1, has four rules:

1. **repr** that takes an opetope in our previous system and makes it into the representable opetopic set of that opetope;
2. **zero** that constructs the empty OCMT;
3. **sum** that takes the disjoint union of two opetopic sets;
4. **glue** that identifies cells of an opetopic set.

In virtue of the fact that every finite opetopic set is a quotient of a finite sum of representables, those rules should be enough to derive all finite opetopic sets. This is proved in theorem 3.6.15 on page 35.

---

**Figure 3.5.1.** The OptSet$^1$ system.

<table>
<thead>
<tr>
<th>Introduction of all targets:</th>
<th>This rule takes an opetope $(E \triangleright \Gamma \vdash_n x : X)$, and completes it by adding all the missing cells, i.e. targets, and making it into an OCMT. If $x \in \forall_n$, then</th>
</tr>
</thead>
<tbody>
<tr>
<td>$E \triangleright \Gamma \vdash_n x : X \quad \text{repr}$</td>
<td>where</td>
</tr>
<tr>
<td>$\Gamma' := \Gamma \cup { t^k a : s^{k+1} a \leftrightarrow s^{k+2} a \leftrightarrow \cdots \mid a \in \forall_{\Gamma,i}, 1 \leq k \leq l \leq n }$,</td>
<td>and</td>
</tr>
</tbody>
</table>
| $E' := E$ | \begin{align*}
& \cup \{ ta = b \mid \text{for all } b \leftrightarrow a(\cdots) \text{ occurring in a type in } \Gamma \} \\
& \cup \{ tta = ts_{\{j\}} s a \mid a \in \forall_{\Gamma,k} \text{ non deg.}, 2 \leq k \leq n \} \\
& \cup \{ t^{k+2} a = b \mid \text{if } t^k a : b \leftrightarrow b \leftrightarrow \cdots, 0 \leq k \leq n-2 \}. 
\end{align*}

Here, $t^k a = t \cdots t a$ is a tagging on the variable $a \in \forall_i$, but for simplicity, we consider it as a variable of its own: $t^k a \in \forall_{I-k}$. By convention, $t^0 a := a$, and if $a = b$, then $ta = tb$, for all $a, b \in \Gamma'$. In line (3.5.2), the source of $a$ is assumed non degenerate, thus $sa$ is a term of the form $x(y_i \leftrightarrow u_i)$, and $s_{\{j\}} sa = x$ (see definition 3.2.1 on page 20).

**Zero:** This rule introduces the empty OCMT.  

---

**Binary sums:** This rule takes two disjoint opetopic sets (i.e. whose cells have different names), and produces their sum. If $\Gamma \cap \Upsilon = \emptyset$, then

$E \triangleright \Gamma \quad F \triangleright \Upsilon \quad \text{sum} \quad \frac{E \triangleright \Gamma \cup F \triangleright \Upsilon}{E \cup F \triangleright \Gamma \cup \Upsilon}$

**Quotients:** This rule identifies two parallel cells in an opetopic set by extending the underlying equational theory. If $a, b \in \forall_{\Gamma}$ are such that $sa = sb$ and $ta = tb$, then

$E \triangleright \Gamma \quad \text{glue} \quad \frac{E \triangleright \Gamma \cup \{ a = b \}}{E \cup \{ a = b \} \triangleright \Gamma}$

We also write \text{glue-}(a=b) to make explicit that we added $\{ a = b \}$ to the theory.

---

**Remark 3.5.4.** Akin to Opt$^1$, in OptSet$^1$, an OCMT that is equivalent to a derivable one is itself derivable.

**Remark 3.5.5.** The sum and zero rules may be replaced by the following usum rule (unbiased sum) without changing the set of derivable OCMTs: for $k \geq 0$, $(E_i \triangleright \Gamma_i), \ldots, (E_k \triangleright \Gamma_k)$ OCMTs such that $\Gamma_i \cap \Gamma_j = \emptyset$ for all $i \neq j$, then

$\sum_{i=1}^k E_i \triangleright \Gamma_i \quad \text{usum} \quad \frac{E_i \triangleright \Gamma_i \quad \ldots \quad E_k \triangleright \Gamma_k}{\sum_{i=1}^k E_i \triangleright \sum_{i=1}^k \Gamma_i}$
3.6. Equivalence with opetopic sets

3.6.1. Opetopic sets from OCTM. Let $E \vdash \Gamma$ and $F \vdash \Upsilon$ be two OCMTs. Then we write $\Gamma/E$ for the set $\forall \Gamma$ quotiented by the equivalence relation generated by the equational theory $E$, and likewise for $\Upsilon/F$.

For $(E \vdash \Gamma \vdash_n x : X)$ a derivable sequent on $\text{Opet1}$, where $x \in \forall_n$, let $(T_x \vdash C_x)$, the $\text{OCMT}$ of $x$, be given by

$$\frac{E \vdash \Gamma \vdash_n x : X}{T_x \vdash C_x} \text{ repr}$$

We wish to prove that $C_x/T_x$ carries a natural structure of representable opetopic set. We proceed in 4 steps:

1. (Sources) If $a \in \forall_{C_x,k}$. If $a$ is not a target, then the sequent $(E|_a \vdash \Gamma|_a \vdash_k a : A)$ is also derivable (where $(-)|_a$ denotes restriction of contexts and theories to variables occurring in type $A$, the type of $a$ in $\Gamma$), and its proof tree is a subtree of that of $x$. Thus we have a well-defined opetope $\llbracket a \rrbracket_k^\text{poly} \in \bigodot_k$. Otherwise, if $a = t^{n-k} x$, then $s a = s^{n-k+1} x$, and define $\llbracket a \rrbracket_k^\text{poly} = \llbracket s^n x \rrbracket_k^\text{poly}$. We thus have a map $\llbracket - \rrbracket_k^\text{poly} : \forall_{C_x} \to \bigodot_k$.

Proposition 3.6.1. The map $\llbracket - \rrbracket_k^\text{poly} : \forall_{C_x} \to \bigodot_k$ factors through $C_x/T_x$.

Proof. By construction, the theory $T_x$ identifies variables $a, b \in \forall_{C_x,k}$ only if $s a = s b$, thus $\llbracket a \rrbracket_k^\text{poly} = \llbracket s a \rrbracket_k^\text{poly} = \llbracket b \rrbracket_k^\text{poly}$. For $\psi \in \bigodot_k$, write $(C_x/T_x)_\psi = \{ a \in \forall_{C_x,k} \mid \llbracket a \rrbracket_k^\text{poly} = \psi \}$.\qed

We now construct source and target maps between those subsets.

1. (Sources) If $[p] \in \llbracket a \rrbracket_k^\text{poly}$, then, by corollary 3.2.13 on page 23, there is a unique $b \in \forall_{C_x\downarrow a,k-1}$ such that $\&_a b = [p]$. Write then $s_{[p]} a = b$.

2. (Target) For $a \in \forall_{C_x,k}$, $k > 0$, we of course set $t(a) = t a$, the latter being a variable introduced by the $\text{repr}$ rule.

Proposition 3.6.2. Let $a \in \forall_{C_x,k}$.

1. For $[p] \in \llbracket a \rrbracket_k^\text{poly}$ we have $\llbracket s_{[p]} a \rrbracket_{k-1} = s_{[p]} \llbracket a \rrbracket_k^\text{poly}$.

2. We have $\llbracket t a \rrbracket_k^\text{poly} = t \llbracket a \rrbracket_k^\text{poly}$.

Proof. 1. Write $[p] = \&_{s a} b$, for some $b \in \llbracket s a \rrbracket^\text{poly}$ (see corollary 3.2.13 on page 23). Then, by corollary 3.2.8 on page 21, we have $s_{[p]} \llbracket a \rrbracket_k^\text{poly} = \llbracket b \rrbracket_{k-1} = s_{[p]} \llbracket a \rrbracket_{k-1}^\text{poly}$.

2. By proposition 3.2.11 on page 22, $t \llbracket a \rrbracket_k^\text{poly} = \llbracket s s a \rrbracket_{k-1}^\text{poly} = \llbracket s t a \rrbracket_{k-1}^\text{poly} = \llbracket t a \rrbracket_{k-1}^\text{poly}$.\qed

Theorem 3.6.3. With all the structure introduced above, $C_x/T_x$ is an opetopic set.

Proof. We check the opetopic identities of section 2.2.4 on page 13. Take $a \in \forall_{C_x,k}$.

1. (Inner) Take $[p[q]] \in \llbracket a \rrbracket_k^\text{poly}$, and write $d = s_{[p[q]]} a$. In $s a$, the variable $d$ occurs as $s a = \ldots$, $b(c \leftrightarrow d)$, ... for some $b \in \star$ and $c \in \llbracket s b \rrbracket^\star$. Then, $[p[q]] = \&_{s a} d = \&_{s b} c \cdot [[\&_{s b} c]]$, and thus $\&_{s b} a = [p]$ and $\&_{s b} c = [q]$. Finally, $s_{[q]} s_{[p]} a = s_{[p[q]]} b = c = t d = t s_{[p[q]]} a$.

2. (Glob1) Assume that $a$ is not degenerate. Then, by definition of $T_x$, we have $t t a = t s_{[1]} a$.\qed
Proposition 3.6.5. (1) We have \( \# \bullet = 1 \), and \( \# \circ = 3 \).

(2) For a non-degenerate opetope \( \omega \in \mathbb{O}_n \), with \( n \geq 2 \), we have

\[
\# \omega = 2 + \left( \sum_{[p] \in \bullet} \# s_{[p]} \omega \right) - \left( \sum_{[q] \in \circ} \# s_{[q]} s_{[p]} \omega \right)
\]

(3.6.6)

(3) If \( \omega \) is a degenerate opetope, say \( \omega = \phi \), then \( \# \omega = 2 + \# \phi \).

Proof. This is a straightforward enumeration exercise from the definition of \( \mathbb{O} \) by generators and relations (see section 2.2.4 on page 13), and using opetopic induction (see remark 2.2.3 on page 12).

Corollary 3.6.7. Let \( \omega \in \mathbb{O}_n \), for \( n \geq 1 \).

(1) If \( \omega = \text{Y}_\psi \) for some \( \psi \in \mathbb{O}_{n-1} \) then \( \# \omega = 2 + \# \psi \).

(2) If \( \omega = \nu \circ [l] \text{Y}_\psi \), for some \( \nu \in \mathbb{O}_n \), \([l] \in \nu \), and \( \psi \in \mathbb{O}_{n-1} \), then

\[
\# \omega = \# \nu + \# \psi - \# \epsilon_{[l]} \nu.
\]

(3.6.8)

Proof. (1) Using proposition 3.6.5, we have

\[
\# \omega = 2 + \left( \sum_{[p] \in \bullet} \# s_{[p]} \omega \right) - \left( \sum_{[q] \in \circ} \# s_{[q]} s_{[p]} \omega \right) = 2 + \# \psi.
\]
(2) Using proposition 3.6.5 on the preceding page, we have
\[
\# \omega = 2 + \left( \sum_{p \in \Omega} \# s_p \omega \right) - \left( \sum_{p \in \Omega} \# s_p s_{p'} \omega \right)
= 2 + \left( \# s_0 + \sum_{p \in \Omega} \# s_p \nu \right) - \left( \# e_0 + \sum_{p \in \Omega} \# s_p s_{p'} \nu \right)
= \# \nu + \# s_0 - \# e_0 \nu.
\]

**Examples 3.6.9.** Consider the opetopic integer \( n \in \mathbb{N} \) from example 3.3.2 on page 26. We show that \( \# n = 2n+3 \). If \( n = 0 \), then by definition, \( \# 0 = \# \mathbf{1} = 2 + \# \Rightarrow = 3 \). If \( n = 1 \), then \( \# 1 = \# Y_\Rightarrow = 2 + \# \Rightarrow = 5 \) by corollary 3.6.7 on the preceding page. Otherwise,
\[
\# n = \# (n - 1) \circ_{\mathbf{1} n-1} Y_\Rightarrow \quad \text{by def. of } n
= \# (n - 1) + \# \Rightarrow - \# e_{1 n-1} (n - 1) \quad \text{by eq. (3.6.8)}
= (2n + 1) + 3 - \# \Rightarrow \quad \text{by ind.}
= (2n + 1) + 3 - 1 = 2n + 3.
\]

As an other example, consider the 3-opetope \( \omega = Y_2 \circ_{\mathbf{1} x} Y_0 \) of example 3.3.5 on page 27:

Then,
\[
\# \omega = \# \left( Y_2 \circ_{\mathbf{1} x} Y_0 \right)
= \# Y_2 + \# 0 - \# e_{1 x} Y_2 \quad \text{by eq. (3.6.8)}
= 2 + \# 2 + \# 0 - \# \Rightarrow \quad \text{by coroll. 3.6.7}
= 9 \quad \text{since } \# n = 2n + 3.
\]

**Proposition 3.6.10.** We have an isomorphism \( C_x/T_x \cong O(\llbracket x \rrbracket_1) \) of opetopic sets.

**Proof.** If \( x = \mathbf{1} \), then \( \forall C_x = \mathbf{1} \), while \( T_x = \emptyset \). Thus \( \# C_x/T_x = 1 = \# \mathbf{1} \) by proposition 3.6.5 on the preceding page. We know by lemma 3.6.4 on the facing page that \( C_x/T_x \) is a quotient of \( O(\llbracket x \rrbracket_1) \), and we just showed that the two have the same number of cells, namely \( \# \mathbf{1} = 1 \). Consequently, \( C_x/T_x \cong O(\llbracket x \rrbracket_{10}) \).

Likewise, if \( x = \Rightarrow \), then \( \forall C_x = \{ \mathbf{1}, \mathbf{1}, x \} \), while \( T_x = \emptyset \). Thus \( \# C_x/T_x = 3 = \# \Rightarrow \) by proposition 3.6.5 on the preceding page, and by the same argument as above, \( C_x/T_x \cong O(\llbracket x \rrbracket_{10}) \).

Assume now that \( x \in \forall_3 \) for \( n \geq 2 \). We proceed by cases on the form of \( sx \).

1. If \( sx = y \in \forall_{n-1} \), then \( \llbracket x \rrbracket_{1 n-1} = Y_{[s]} \llbracket y \rrbracket_{1 n-1} \) so that \( \# \llbracket x \rrbracket_{1 n} = 2 + \# \llbracket y \rrbracket_{1 n-1} \) by proposition 3.6.5 on the facing page. Then \( C_x = C_y + \{ t^k x \mid 0 \leq k \leq n \} \), and \( t x T_x t s \{ x = t y \} \). Consequently, \( T_x \) is equivalent to the theory \( T_y + \{ t x = t y \} \), and thus
\[
C_x/T_x = C_y/T_y + \{ x, t x \}.
\]

By induction, \( C_y/T_y \cong O(\llbracket y \rrbracket_{1 n-1}) \), and \( \# C_x/T_x = 2 + \# C_y/T_y = 2 + \# \llbracket y \rrbracket_{1 n-1} = \# \llbracket x \rrbracket_{1 n} \), which, by the same argument as above, proves the isomorphism \( C_x/T_x \cong O(\llbracket x \rrbracket_{1 n}) \).

2. If \( sx = a \) for some \( a \in \forall_{n-2} \), then \( \llbracket x \rrbracket_{1 n} = \llbracket a \rrbracket_{1 n-2} \) so that \( \# \llbracket x \rrbracket_{1 n} = 2 + \# \llbracket a \rrbracket_{1 n-2} \) by proposition 3.6.5 on the preceding page. Then \( C_x = C_a + \{ t^k x \mid 0 \leq k \leq n \} \), and \( t x T_x a \). Therefore \( T_x \) is equivalent to the theory \( T_a + \{ t x = a \} \), and thus
\[
C_x/T_x = C_a/T_a + \{ x, t x \}.
\]

Consequently, \( \# C_x/T_x = 2 + \# C_a/T_a = 2 + \# \llbracket a \rrbracket_{1 n-2} = \# \llbracket x \rrbracket_{1 n} \).
3. NAMED APPROACH

(3) Assume \( s x = t(a \leftarrow y) \), for some \( t \in \mathbb{T}_{n-1} \), \( a \in \mathbb{V}_{n-2} \), and \( y \in \mathbb{V}_{n-1} \). Let \( z : t \mapsto \ldots \) be a fresh \( n \)-variable. Since all cells of \( z \) except \( z \) and its targets are also cells of \( x \), we have

\[
C_x = C_y \cup \left( C_z - \{ t^k z \mid 0 \leq k \leq n \} \right) + \{ t^k x \mid 0 \leq k \leq n \},
\]

while \( T_x \) is equivalent to \( T_y \cup T_z + \{ ty = a, tt x = ts_z [x] \} \). Since \( s_z [x] = s_z [z] \), and \( ts_z [x] z T_z tt z \), we have

\[
C_x / T_x = C_y / T_y \cup C_z / T_z + \{ x, tx \} - \{ z, tz \}.
\]

By hypothesis of the graft rule, \( C_z / T_z \cap C_y / T_y = C_a / T_a \), thus

\[
\# \sum C_x / T_x = \# \sum C_y / T_y + \# \sum C_z / T_z - \# \sum C_a / T_a = \# \sum \# \sum \# \sum a_{\text{poly}} \sum b_{\text{poly}} - \# \sum a_{\text{poly}} \sum b_{\text{poly}}
\]

\[
= \# \left( \sum \# \sum \# \sum a_{\text{poly}} \sum b_{\text{poly}} \bigg| \sum a_{\text{poly}} \sum b_{\text{poly}} \right).
\]

Let \( (E \triangleright \Gamma) \) be a derivable OCMT in \( \text{OptSet}^{\dagger} \), \( a \in \mathbb{V}_{\Gamma,k} \), and \( (E|_a \triangleright \Gamma|_a) \) the restriction to \( a \). Then we have an inclusion \( i : \Gamma|_a / E|_a \rightarrow \Gamma / E \), where \( i : b \mapsto b \) for all \( b \in \mathbb{V}_{\Gamma|_a} \), and thus we have a natural map \( \tilde{a} : O[\sum a_{k_{\text{poly}}}^{\text{poly}}] \rightarrow \Gamma / E \) given by the composite

\[
O[\sum a_{k_{\text{poly}}}^{\text{poly}}] \xrightarrow{\tilde{a}} \sum a_{\text{poly}} / E |_a \rightarrow \sum a_{\text{poly}} / E \rightarrow \sum a_{\text{poly}} / E |_a \rightarrow \Gamma / E,
\]

where the middle map comes from the fact that \( C_a = \sum a_{\text{poly}} \) up to renaming, and that \( T_a \in E |_a \). Let now \( (E \triangleright \Gamma) \) and \( (F \triangleright \Upsilon) \) be two OCMTs, and assume by induction that \( \Gamma / E \) and \( \Upsilon / F \) have a structure of opetopic set. Then, by definition of rule sum, and for \( (G \triangleright \Xi) \) given by

\[
\begin{array}{c}
E \triangleright \Gamma \\
G \triangleright \Xi \\
\sum
\end{array}
\rightarrow
\begin{array}{c}
F \triangleright \Upsilon \\

\end{array}
\]

we have \( \Xi = \Gamma + \Upsilon \), and \( G = E + F \). We endow the quotient \( \Xi / G \) with a structure of opetopic set as follows:

\[
\Xi / G := \Gamma / E + \Upsilon / F.
\]

(3.6.11)

Let \( a, b \in \mathbb{V}_{\Gamma,k} \) be such that \( s a = E s b \) and \( t a = E t b \). Remark that \( O[\sum a_{k_{\text{poly}}}^{\text{poly}}] = O[\sum b_{k_{\text{poly}}}^{\text{poly}}] \). Then, by definition of rule glue, and for \( (F \triangleright \Gamma) \) given by

\[
\begin{array}{c}
E \triangleright \Gamma, F \triangleright \Gamma \\
\text{glue}\ (a = b)
\end{array}
\rightarrow
\begin{array}{c}
\Gamma / F \\

\end{array}
\]

we have \( F = E + \{ a = b \} \). We endow the quotient \( \Gamma / F \) with a structure of opetopic set defined by the following coequalizer:

\[
\begin{array}{c}
O[\sum a_{k_{\text{poly}}}^{\text{poly}}] \xrightarrow{\tilde{a}} \Gamma / E \\
\tilde{b} \rightarrow \Gamma / F;
\end{array}
\]

(3.6.12)

Proposition 3.6.13. For \( (E \triangleright \Gamma) \) a derivable OCMT in \( \text{OptSet}^{\dagger} \), the structure of opetopic set on \( \Gamma / E \) does not depend on the proof tree of \( (E \triangleright \Gamma) \).

Proof. The opetopic set \( \Gamma / E \) is given by the following expression that does not depend on the proof tree of \( (E \triangleright \Gamma) \):

\[
\Gamma / E \equiv \sum_{k \in \mathbb{N}, a \in \mathbb{V}_{\Gamma,k}} O[\sum a_{k_{\text{poly}}}^{\text{poly}}] / a \sim b, \forall a, b \in \mathbb{V}_{\Gamma} \text{ s.t. } a = E b.
\]

Indeed, it holds when \( (E \triangleright \Gamma) \) is of the form \( (T_a \triangleright C_a) \), and since other non-empty OCMTs are given by application of rules sum and glue, we can show by induction on the proof tree of \( (E \triangleright \Gamma) \) that the opetopic set \( \Gamma / E \) doesn’t depend on that proof tree! \( \square \)
3.6.2. Equivalence. Recall that $\mathcal{O}$ is the category of opetopic sets, and that $\mathcal{Fin}\mathcal{O}$ is the full subcategory of $\mathcal{O}$ spanned by finite opetopic sets (see section 2.2.4 on page 13). In this subsection, we provide the last results needed to establish the equivalence between the category of derivable OCMTs and $\mathcal{Fin}\mathcal{O}$.

For $\Gamma \ast \Gamma$ an OCMT, and $a, b \in V$, the substitution $\Gamma[a/b]$ is defined in the obvious manner, by applying said substitution to all typings in $\Gamma$. A morphism $f : (\Gamma \ast \Gamma) \to (\Gamma \ast \mathcal{Y})$ is a (non necessarily bijective) map $f : \Gamma \to \mathcal{Y}$ compatible with $E$ and $F$, such that if $x : X$ is a typing in $\Gamma$, then $f(x) : f(X)$ is a typing in $\mathcal{Y}$, where $f(X)$ is the result of applying $f$ to every variable in $X$. Note that this condition implies that $f$ preserves the dimension of variables. Also, if $f, g : (\Gamma \ast \Gamma) \to (\Gamma \ast \mathcal{Y})$, and if for all $x \in \Gamma$ we have $f(x) =_{F} g(x)$, then $f = g$. We write $\text{Ctx}^\prime$ for the category of derivable OCMTs and such morphisms. In a sense, it is the syntactic category of the $\text{OFS}$ system.

**Lemma 3.6.14.** Let $f : (\Gamma \ast \Gamma) \to (\Gamma \ast \mathcal{Y})$ be a morphism of OCMT, and $a \in \Gamma, k$. Then $[a]^{\text{poly}}_{k} = [f(a)]^{\text{poly}}_{k}$. Informally, morphisms of OCMTs preserve the “shape” of variables.

**Proof.** Since there is a unique 0-opetope and a unique 1-opetope, the result holds trivially if $k = 0, 1$. Assume now $k \geq 2$. We proceed by induction on $k$.

1. If $a = b \in \Gamma_{k-1}$, then $[a]^{\text{poly}}_{k} = \mathcal{Y}[b]^{\text{poly}} = \mathcal{Y}[f(b)]^{\text{poly}} = [f(a)]^{\text{poly}}_{k}$.

2. If $a = b$ for some $b \in \Gamma_{k-2}$, then $[a]^{\text{poly}}_{k} = \mathcal{Y}[b]^{\text{poly}} = \mathcal{Y}[f(b)]^{\text{poly}} = [f(a)]^{\text{poly}}_{k}$.

3. If $a = b(c_{i} \leftarrow u_{i})$, we can show by a secondary induction that $[u_{i}]^{\text{poly}}_{k} = [f(u_{i})]^{\text{poly}}_{k}$, and $[a]^{\text{poly}}_{k} = \mathcal{Y}[b]^{\text{poly}}_{k} \bigcirc \mathcal{Y}[u_{i}]^{\text{poly}}_{k} = [f(a)]^{\text{poly}}_{k}$.

The named stratification functor $S^{1} : \text{Ctx}^\prime \to \mathcal{Fin}\mathcal{O}$ is defined as follows:

$$S^{1} : \text{Ctx}^\prime \to \mathcal{Fin}\mathcal{O}$$

$$E \ast \Gamma \to \Gamma/E$$

$$\left( \left( \Gamma \ast \Gamma \right) \xrightarrow{f} (\Gamma \ast \mathcal{Y}) \right) \to \left( \Gamma/E \xrightarrow{f} \mathcal{Y}/F \right).$$

By lemma 3.6.14, $f = S^{1}(E \ast \Gamma) \to S^{1}(\Gamma \ast \mathcal{Y})$ is indeed a morphism of opetopic sets.

**Theorem 3.6.15.** The stratification functor $S^{1} : \text{Ctx}^\prime \to \mathcal{Fin}\mathcal{O}$ is an equivalence of categories.

**Proof.** The category $\text{im}D$, consisting of finite opetopic sets of the form $\Gamma/E$ for some OCMT $E \ast \Gamma$, contains all representables (proposition 3.6.10 on page 33), the initial object (since $S^{1}(\ast)$ is the opetopic set with no cell), and is closed under finite sums (equation (3.6.11) on the preceding page) and quotients (equation (3.6.12) on the facing page). Thus its closure under isomorphism is finitely cocomplete, and equal to the whole category $\mathcal{Fin}\mathcal{O}$, so $S^{1}$ is essentially surjective. By definition, $S^{1}$ is also faithful, and it remains to show that it is full.

Let $f : S^{1}(E \ast \Gamma) \to S^{1}(F \ast \mathcal{Y})$ be a morphism of opetopic sets. Then, in particular, it is a map $f : \Gamma/E \to \mathcal{Y}/F$ between the set of cells of $S^{1}(E \ast \Gamma)$ and $S^{1}(F \ast \mathcal{Y})$. To prove that it is a morphism of OCMT, we show that $\Gamma[f(x)/x \mid x \in \Gamma]$ is a subcontext of $\mathcal{Y}$ modulo $F$, i.e. that for all typings $x : X$ in $\Gamma$, for some $x \in \Gamma_{k}$, the type of $f(x)$ in $\mathcal{Y}$ is $f(X)$ modulo $F$. If $\Gamma \ast \Gamma$ is the empty OCMT, the result is trivial, so we assume now that this is not the case. Since $f$ is a morphism of opetopic sets, we have $f(x) \in \mathcal{Y}_{X, k, t}$, and by lemma 3.6.14, $[x]^{\text{poly}}_{k} = [f(x)]^{\text{poly}}_{k}$. We proceed by induction on $k$.

1. If $k = 0$, then $X = \emptyset$. Since $f(x) \in \mathcal{Y}_{X, 0}$, its type is necessarily $\emptyset = f(X)$, thus $f(x) : f(X)$ is a typing in $\mathcal{Y}$.

2. If $k = 1$, then $X = (s_{1}x \leftrightarrow \emptyset)$, and, since $f$ is a morphism of opetopic sets, $f(s_{1}x) =_{F} s_{1}f(x)$. Thus $f(X) = (f(s_{1}x) \leftrightarrow \emptyset) =_{F} (s_{1}f(x) \leftrightarrow \emptyset)$, the latter being the type of $f(x)$ in $\mathcal{Y}$. 

3.6. EQUIVALENCE WITH OPETOPIC SETS
(3) Assume now that \(k \geq 2\). The type \(X\) of \(x\) in \(\Gamma\) is by definition \(X = (s x \multimap ssx \multimap \ldots \multimap \varnothing)\), and the type of \(t x\) in \(\Gamma\) is \(Y := (ssx \multimap \ldots \multimap \varnothing)\). By induction, the type of \(f(t x)\) in \(\Upsilon\) is \(f(Y)\), and since \(f(t x) = t f(x)\), and the type of the latter is by definition \(ssf(x) \multimap \ldots \multimap \varnothing\), we have

\[
(f(ssx) \multimap \ldots \multimap \varnothing) = f(Y) = f_s(ssf(x) \multimap \ldots \multimap \varnothing),
\]

or in other words, \(s^i f(x) = f(s^i x)\), for \(2 \leq i \leq k\). It remains to show that the latter formula holds in the case \(i = 1\) (the case \(i = 0\) is tautological). Towards a contradiction, assume \(sf(x) \neq f(s x)\). Then there exists \([p] \in \mathbb{x}_{\text{poly}}^* \equiv \mathbb{x}_k_{\text{poly}}^*\) such that \(s[p] f(x) \neq f(s[p] x)\), which contradicts the fact that \(f\) is a morphism of opetopic sets. Consequently, \(sf(x) \neq f(s x)\), and \(f(X)\) is the type of \(f(x)\) in \(\Upsilon\) modulo \(F\).

Finally, the underlying map \(f : \Gamma/E \to \Upsilon/F\) of the morphism of opetopic sets \(f : S^i(E \triangleright \Gamma) \to S^i(F \triangleright \Upsilon)\) is a morphism of OCMT, and \(S^i\) is full.

**Proposition 3.6.16.** The category \(\text{Ctx}^i\) has finite colimits, and \(S^i\) preserves them.

*Proof*. The empty OCMT is certainly an initial object of \(\text{Ctx}^i\) (it is in fact the only one), and the OCMT \((E + F \triangleright \Gamma + \Upsilon)\) obtained by the rule \(\sum\) is clearly a coproduct of \((E \triangleright \Gamma)\) and \((F \triangleright \Upsilon)\). Let now \(f, g : (E \triangleright \Gamma) \to (F \triangleright \Upsilon)\) be two parallel morphism in \(\text{Ctx}^i\). Consider the following:

\[
(F \triangleright \Upsilon) \xrightarrow{x \mapsto f(x)} (F \cup \{f(x) = g(x) \mid x \in \forall \Gamma\} \triangleright \Upsilon).
\]

Then, \(G \triangleright \Upsilon\) is derived from \(F \triangleright \Upsilon\) by repeated application of the \textit{glue} rule, and by universal property, it is easy to see that it is a coequalizer of \(f\) and \(g\) in \(\text{Ctx}^i\). So, to summarize, \(\text{Ctx}^i\) contains all finite sums and coequalizers, and it is thus finitely cocomplete. The fact that \(S^i\) preserves finite colimits is trivial in the case of initial objects, a consequence of equation (3.6.11) on page 34 for binary sums, and of equation (3.6.12) on page 34 for coequalizers. \(\square\)

**Theorem 3.6.17.** We have an equivalence \(\hat{\mathcal{C}} \simeq \text{Mod}((\text{Ctx}^i)^{\text{op}}, \text{Set})\), where the latter is the category of \((\text{Ctx}^i)^{\text{op}}\)-models in \(\text{Set}\), i.e. finitely continuous functors \((\text{Ctx}^i)^{\text{op}} \to \text{Set}\) and natural transformations\(^2\).

*Proof*. We apply Gabriel–Ulmer duality [Adámek and Rosický, 1994, Lack and Power, 2009]. The category \(\hat{\mathcal{C}}\) is finitely locally presentable as it is the presheaf category of a small category. Finite opetopic sets are the finitely presentable objects, thus

\[
\text{Mod}((\text{Ctx}^i)^{\text{op}}, \text{Set}) \simeq \text{Mod}(\text{Fin}\hat{\mathcal{C}}^{\text{op}}, \text{Set}) \simeq \hat{\mathcal{C}}.
\]

\(\square\)

### 3.7. Examples

In this section, we give example derivations in system OptSet\(^i\). For clarity, we do not repeat the type of previously typed variables in proof trees.

**Example 3.7.1.** The opetopic set

\[
\begin{array}{c}
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ \\
\circ
and applying the \texttt{repr} rule yields respectively:

\[
\begin{align*}
\triangleright a: \emptyset, f: a \leftrightarrow \emptyset, \alpha: f \leftrightarrow \emptyset, t f: \emptyset, t \alpha: a \leftrightarrow \emptyset, t t \alpha: \emptyset \\
\triangleright c: \emptyset, g: c \leftrightarrow \emptyset, t g: \emptyset \\
\triangleright b: \emptyset, h: b \leftrightarrow \emptyset, t h: \emptyset.
\end{align*}
\]

The proof tree then reads:

\[
\begin{array}{c}
\vdash a; f; \alpha \\
\vdash c; g; \alpha \sum \\
\vdash b; h; \alpha \sum \\
\vdash t t = t f ; g; h; \alpha \sum \\
\vdash g = t \alpha \sum
\end{array}
\]

\[\text{Example 3.7.2.} \text{ The opetopic set}
\]

\[
\begin{tikzpicture}
\begin{scope}[nodes={draw, circle, inner sep=0pt, minimum size=2pt}]
\node (h) at (0,0) {}; \\
\node (g) at (1,0) {}; \\
\node (f) at (0,1) {}; \\
\node (a) at (0,-1) {}; \\
\node (b) at (1,1) {};
\end{scope}
\draw (h) -- (g) -- (f) -- (a) -- (h); \\
\node at (0,0) {$a$}; \\
\node at (1,0) {$b$}; \\
\node at (0,1) {$f$}; \\
\node at (0,-1) {$a$}; \\
\node at (1,1) {$b$}; \\
\end{tikzpicture}
\]

is derived as follows. First, we derive the cells \(\alpha\) and \(\beta\) as opetopes to obtain the following sequents:

\[
\begin{align*}
\triangleright a: \emptyset, b: \emptyset \\
\triangleright f: a \leftrightarrow \emptyset, g: a \leftrightarrow \emptyset \\
\triangleright_2 \alpha: g(b \leftrightarrow f) \leftrightarrow a \leftrightarrow \emptyset \\
\triangleright \alpha: g(b \leftrightarrow f) \leftrightarrow a \leftrightarrow \emptyset \\
\triangleright \alpha: g(b \leftrightarrow f) \leftrightarrow a \leftrightarrow \emptyset \\
\triangleright h: a \leftrightarrow \emptyset \\
\triangleright \beta: h \leftrightarrow a \leftrightarrow \emptyset
\end{align*}
\]
Applying the \texttt{repr} rule yields respectively:

\begin{align*}
& b = tf, tg = tt_\alpha \\
& a : \emptyset, b : \emptyset, tf : \emptyset, tg : \emptyset, tt_\alpha : \emptyset \\
& f : a \leftrightarrow \emptyset, g : a \leftrightarrow \emptyset, tt_\alpha : a \leftrightarrow \emptyset \\
& \alpha : g(b \leftrightarrow f) \leftrightarrow a \leftrightarrow \emptyset \\
& a' : \emptyset, th : \emptyset, tt_\beta : \emptyset \\
& th = tt_\beta \\
& h : a' \leftrightarrow \emptyset, tt_\beta : a' \leftrightarrow \emptyset \\
& \beta : h \leftrightarrow a' \leftrightarrow \emptyset
\end{align*}

The proof tree then reads:

\begin{align*}
\text{glue-(a = tg)} \\
& b = tf, a = tg = tt_\alpha \\
& a, b, tf : \emptyset, tg : \emptyset, tt_\alpha : \emptyset \\
& f, g, ta : a \leftrightarrow \emptyset \\
& \alpha : a, b, tf, tg, tt_\alpha \\
& \text{sum} \\
& b = tf, a = tg = tt_\alpha, a' = th = tt_\beta \\
& h = t_\beta \\
& a, b, tf, tg, tt_\alpha, a', th, tt_\beta \\
& \alpha, \beta \\
& \text{glue-(a = a')} \\
& b = tf, a = tg = tt_\alpha = a' = th = tt_\beta \\
& h = t_\beta \\
& a, b, tf, tg, tt_\alpha, a', th, tt_\beta \\
& \alpha, \beta \\
& \text{glue-(ta = h)} \\
& b = tf, a = tg = tt_\alpha = a' = th = tt_\beta \\
& h = t_\beta = ta \\
& a, b, tf, tg, tt_\alpha, a', th, tt_\beta \\
& \alpha, \beta \\
& \text{sum} \\
& b = tf, a = tg = tt_\alpha = th = tt_\beta \\
& h = t_\beta \\
& a, b, tf, tg, tt_\alpha, a', th, tt_\beta \\
& \alpha, \beta
\end{align*}

Write \((E \triangleright \Gamma)\) for the final OCMT of this proof tree. At the beginning of section 3.5.1 on page 29, we gave a slightly different OCMT for the same opetopic set:

\begin{align*}
& (F \triangleright \Upsilon) := \\
& (b = tf, a = tg = tt_\alpha = th = tt_\beta \\
& h = t_\beta = ta \\
& a : \emptyset, b : \emptyset, tf : \emptyset, tg : \emptyset, tt_\alpha : \emptyset, th : \emptyset, tt_\beta : \emptyset \\
& f : a \leftrightarrow \emptyset, g : a \leftrightarrow \emptyset, ta : a \leftrightarrow \emptyset, h : a \leftrightarrow \emptyset, tt_\beta : a \leftrightarrow \emptyset \\
& \alpha : g(b \leftrightarrow f) \leftrightarrow a \leftrightarrow \emptyset, \beta : h \leftrightarrow a \leftrightarrow \emptyset)
\end{align*}

However, there exists an isomorphism \(f : (E \triangleright \Gamma) \rightarrow (F \triangleright \Upsilon)\), defined by mapping \(a'\) to \(a\), and variables in \(\mathcal{V}_\Gamma - \{a'\}\) to those in \(\mathcal{V}_\Upsilon\) with the same name. Its inverse it given by mapping \(a\) to either \(a\) or \(a'\), and similarly for the over variables of \(\Upsilon\).

### 3.8. Python implementation

The system \texttt{OptSet} is implemented in module \texttt{opetopy.NamedOpetopicSet} of [Ho Thanh, 2018b]. The rules are represented by functions \texttt{repr} (since \texttt{repr} a Python standard function), \texttt{sum}, \texttt{glue}, and \texttt{zero}, and are further encapsulated in rule instance classes \texttt{Repr}, \texttt{Sum}, \texttt{Glue}, and \texttt{Zero}. We do not discuss the implementation, but we give an example derivation in figure 3.8.1 on the next page.
from opetopy.NamedOpetopicSet import *

# We first define all relevant variables using system Opt^!.

f = Shift(Point("a"), "f")
g = Shift(Point("c"), "g")
h = Shift(Point("b"), "h")
alpha = Shift(f, "alpha")

# We then take the sum of all the representables we need. Note that the new target variables added by the repr rule have "t" prepended to their name e.g. the target variable of f is "tf", while that of alpha is "talpha".

element_unglued = Sum(Sum(Repr(alpha), Repr(g)), Repr(h))
element = Glue(Glue(Glue(Glue(Glue(element_unglued,
                                                        "a",
                                                        "b",
                                                        "b",
                                                        "a",
                                                        "g",
                                                        "th"),
                                                        "c"),
                                                        "tf"),
                                                        "tg"),
                                                        "tf")

3.9. The mixed system for opetopic sets

The OptSet^! system, presented in section 3.5 on page 29, suffers from the following drawback: derivations of opetopic sets start with instances of rules zero or repr, the latter requiring a full opetope derivation in system Opt^! (presented in section 3.1 on page 15). This makes derivations somewhat unintuitive, since for an opetopic set X ∈ Õ written as

\[ X = \sum_i \mathcal{O}[\omega_i] \]

where ~ represents some quotient, the opetopes \( \omega_i \) have to be derived in Opt^! first, then the repr rule has to be used on each one to produce the corresponding representables \( \mathcal{O}[\omega_i] \), and only then can the sums and gluing be performed.

In this section, we present system OptSet^!_m (the m standing for “mixed”) for opetopic sets, which does not depend on Opt^!, and allows to perform introductions of new cells, sums, and gluings in any sound order. This is done by introducing new cells along with all their targets, effectively rendering OptSet^!'s repr rule superfluous, and removing the “barrier” between Opt^! and OptSet^! in the above schema.

**3.9.1. Syntax.** The syntax of system OptSet^!_m uses sequents from Opt^! (see section 3.1.1 on page 15) and OCMTs from OptSet^! (see section 3.5.1 on page 29). Specifically, we use two types of judgments.

1. \( E \vdash \Gamma \), stating that \( E \vdash \Gamma \) is a well formed OCMT.
2. \( E \vdash \Gamma \vdash t : T \), stating that in OCMT \( E \vdash \Gamma \), the term \( t \) is well formed, and has type \( T \). We may also write \( E \vdash \Gamma \vdash t : T \) if \( t \in \mathbb{T}_n \).

**3.9.2. Inference rules.** We present the inference rules of system OptSet^!_m in figure 3.9.1 on the following page. It uses rules point, degen, and graft from system Opt^!, and rules zero, sum, and glue from system OptSet^!.

Rule pd is a counterpart to degen, introducing a non degenerate pasting diagram from an OCMT. Lastly, rule shift is a variant of that of system Opt^!, and introduces a new cell from a pasting diagram, along with all its targets. It can be viewed as a fusion of Opt^!'s shift rule and OptSet^!'s repr rule.
Figure 3.9.1. The $\text{OptSet}^1_\Gamma$ system.

**Introduction of points:** This rule introduces 0-cells, also called points. If $x \in \mathbb{V}_0$, then
\[
\vdash x : \emptyset \text{ point}
\]

**Introduction of degenerate pasting diagrams:** This rule creates a new degenerate pasting diagram. If $x \in \mathbb{V}_{\Gamma,k}$, then
\[
E \vdash \Gamma, x : \overset{k}{X} \text{ degen}
\]

**Introduction of non degenerate pasting diagrams:** This rule creates a new non-degenerate pasting diagram consisting of a single cell. It can then be extended using the graft rule. If $x \in \mathbb{V}_{\Gamma,k}$, then
\[
E \vdash \Gamma, x : \overset{k}{X} \text{ pd}
\]

**Grafting:** This rule extends a previously derived non degenerate pasting diagram by grafting a cell. With the same conditions as rule graft of system Opt$^1$ (see section 3.1 on page 15): for $x \in \mathbb{V}_n, t \in \mathbb{T}_n$ is not degenerate, $a \in (st)^*$ such that $sa = \emptyset x$, then
\[
\begin{align*}
E \vdash \Gamma, t : s_1 \leftrightarrow s_2 \leftrightarrow \cdots \quad F \vdash \Upsilon, x : \overset{n}{X} \\
G \vdash \Gamma \cup \Upsilon, t(a \leftarrow x) : s_1[a_1x_1/a] \leftrightarrow s_2 \leftrightarrow \cdots \\
\text{graft}
\end{align*}
\]

where $G$ is the union of $E, F$, and potentially a set of additional equalities incurred by the substitution $s_1[a_1x_1/a]$. We also write graft-$a$ to make explicit that we grafted onto $a$.

**Shifting of pasting diagrams:** This rule takes a previously derived pasting diagram (degenerate or not), and introduces a new cell having this pasting diagram as source. It also introduces the targets of all its iterated sources, and extends the ambient equational theory with the required identities. If $x \notin \mathbb{V}_\Gamma$, then
\[
E \vdash \Gamma, t : T \quad F \vdash \Upsilon \quad \text{shift}
\]

where
\[
\Upsilon := \Gamma \uplus \{ x : t \leftrightarrow T \} \cup \{ t^i : s^{i+1} x \leftrightarrow s^{i+2} x \leftrightarrow \cdots \ | 0 < i \leq n \}
\]

by convention, we let $t^0 x = x$, and where $F$ is defined as follows:

1. If $t$ is a degenerate term, say $t = a$, then
   \[
   F := E \uplus \{ t^i x = t^i a \ | 0 \leq i \leq n-1 \}
   \]
   (3.9.1)

2. If $t$ is not degenerate, say $t = y(\overline{z_i} \leftarrow \overline{u_i})$, for some $y \in \mathbb{V}_n$, $\overline{z_i} \in \mathbb{V}_{n-1}$, and $\overline{u_i} \in \mathbb{T}_n$, then
   \[
   F := E \uplus \{ t^2 x = t y \ | \ n \geq 1 \} \cup \{ t a = b \ | \ \text{for all } b \leftarrow a(\cdots) \text{ occurring in } t \}.
   \]

**Zero:** This rule introduces the empty OCMT.
\[
\vdash \emptyset \text{ zero}
\]

**Binary sums:** This rule takes two disjoint OCMTs (i.e. whose cells have different names), and produces their sum. If $\Gamma \cap \Upsilon = \emptyset$, then
\[
E \vdash \Gamma \quad F \vdash \Upsilon \quad \text{sum}
\]

**Quotients:** This rule identifies two parallel cells in an opetopic set by extending the underlying equational theory. If $a, b \in \mathbb{V}_{\Gamma}$ are such that $sa = Eb$ and $ta = Eb$, then
\[
E \vdash \Gamma \quad \text{glue}
\]

We also write glue-$a=b$ to make explicit that we added $\{a = b\}$ to the theory.
Remark 3.9.2. The \textit{sum} and \textit{zero} rules may be replaced by the following \textit{usum} rule (unbiased sum) without changing the set of derivable OCMTs: for $k \geq 0$, $(E_1 \triangleright \Gamma_1), \ldots, (E_k \triangleright \Gamma_k)$ OCMTs such that $\Gamma_i \cap \Gamma_j = \emptyset$ for all $i \neq j$, then

\[
\frac{E_1 \triangleright \Gamma_1 \quad \ldots \quad E_k \triangleright \Gamma_k}{\sum_{i=1}^k E_i \triangleright \sum_{i=1}^k \Gamma_i} \text{ \textit{usum}}
\]

Remark 3.9.3. Akin to $\text{Opt}^!$ and $\text{OptSet}^!$, in $\text{OptSet}^!_m$ a sequent or an OCMT that is equivalent to a derivable one is itself derivable.

3.10. Equivalence with opetopic sets

The aim of this section is to prove theorem 3.10.4 on page 44, stating that system $\text{OptSet}^!_m$ precisely derives opetopic sets, in the sense of theorems 3.6.15 and 3.6.17 on page 35 and on page 36. In other words, we prove that the set of derivable OCMTs of systems $\text{OptSet}^!_m$ and $\text{OptSet}^!$ are the same. This is done by rewriting proof trees in $\text{OptSet}^!$ to proof trees in $\text{OptSet}^!_m$ (see proposition 3.10.1) and conversely (see proposition 3.10.3 on page 44).

Throughout this section, the rules of systems $\text{Opt}^!$ and $\text{OptSet}^!$ will be decorated by a prime, e.g. $\text{shift}'$, in order to differentiate them from the rules of system $\text{OptSet}^!_m$. Further, to make notations lighter and the demonstrations more graphical, we write proof trees as actual trees, whose nodes are decorated by rules, and edges by sequents or OCMTs. For instance, derivation of the arrow $\triangleright$ (see example 3.3.1 on page 26) in system $\text{Opt}^!$ is represented as on the left, and more concisely as on the right:

\[
\begin{array}{c}
\triangleright \vdash x : \emptyset \triangleright x : \emptyset \\
\triangleright \vdash x : \emptyset, f : \emptyset \triangleright \emptyset \triangleright f : \emptyset \triangleright \emptyset
\end{array}
\]

If no uncertainty arise, we leave the decoration of the edges implicit, as on the right.

**Proposition 3.10.1.** Every OCMT derivable in system $\text{OptSet}^!$ is also derivable in system $\text{OptSet}^!_m$.

**Proof.** Recall that a proof tree in system $\text{OptSet}^!$ has the following structure:

\[
\begin{array}{c}
\text{Opt}^! \\
\text{repr}'
\end{array}
\]

meaning that it begins with derivations in system $\text{Opt}^!$, followed by instances of the \textit{repr}' rule, followed by a derivation in system $\text{OptSet}^!$. Remark that rule \textit{glue}' is exactly \textit{glue}, and likewise for \textit{sum}, so that the bottom part of the proof tree is already a derivation in system $\text{OptSet}^!_m$.

We now show that we can rewrite the top part to a proof in system $\text{OptSet}^!_m$ by “lifting” the instances of \textit{repr}, and replacing the other rule instances by those of $\text{OptSet}^!_m$. This rewriting procedure is defined by the following cases.

1. If we have a proof tree as on the left, we rewrite it as on the right:

\[
\begin{array}{c}
\text{point}' \\
\text{repr}'
\end{array}
\rightarrow
\begin{array}{c}
\text{point}
\end{array}
\]
(2) Likewise, if we have a derivation as on the left, where $\Pi$ is a proof tree in system $\text{Opt}^i$ or $\text{OptSet}^i$, we rewrite it as on the right:

\[
\begin{align*}
\Pi & \quad \Pi \\
\text{degen}'-x & \quad \text{degen}-x \\
\text{shift}'-\delta & \quad \text{shift}-\delta \\
\text{repr}' & \quad \text{repr}'
\end{align*}
\]

(3) Likewise, if we have a proof tree as on the left of figure 3.10.1, we rewrite it as on the right.

\[\text{Figure 3.10.1. Proof of proposition 3.10.1 on the preceding page: commuting \text{repr}' with a \text{graft}' and \text{shift}' sequence}\]

It is routine verification to check that the conclusion OCMT on the left and the right of any of those cases are the same. This rewriting procedure is convergent (i.e. confluent and terminating), and the normal form of a proof tree in system $\text{OptSet}^i$ is a proof tree in system $\text{OptSet}^i_m$ that derives the same OCMT. □

**Lemma 3.10.2.** Let $(E \Rightarrow \Gamma)$ be a derivable OCMT in system $\text{OptSet}^i_m$. Then it admits a proof tree of the following form

\[\text{point, degen, pd, graft, shift}\]

meaning a proof tree starting with derivation in the fragment of system $\text{OptSet}^i_m$ containing only rules point, degen, pd, graft, and shift, followed by a derivation in the complementary fragment.

**Proof.** If we have a proof tree consisting only of an instance of rule zero, then the result trivially holds. Otherwise, we proceed by stating rewriting steps of proof trees in system $\text{OptSet}^i_m$, as in the proof of proposition 3.10.1 on the previous page.
(1) If we have a proof tree as on the left, we rewrite it as on the right:

(2) Likewise, consider a proof tree as on the left. Then, by assumption on rule \textit{sum}, either \( x \in \mathcal{V}_{\Gamma,1} \) or \( x \in \mathcal{V}_{\Gamma,2} \). Without loss of generality, assume the latter holds. Then we rewrite the proof tree as on the right:

(3) Likewise, consider a proof tree as on the left of figure 3.10.2. Then, by assumption on rule \textit{sum}, either \( a_{i-1} \in \mathcal{V}_{\Gamma,1} \) or \( a_{i-1} \in \mathcal{V}_{\Gamma,2} \). Without loss of generality, assume the latter holds. Then we rewrite the proof tree as on the right of the figure.

(4) Likewise, consider a proof tree as on the left of figure 3.10.3 on the next page. Then for \( 1 \leq j \leq k \), we have \( x \in \mathcal{V}_{\Gamma_j} \) if and only if \( y \in \mathcal{V}_{\Gamma_j} \). Assume \( x, y \in \mathcal{V}_{\Gamma_j} \), and by induction, assume that the last rule instance of \( \Pi_j \) is \textit{glue}-(\( x=y \)). Then we rewrite the proof tree as on the right of the figure, where \( \Pi_j' \) is \( \Pi_j \) with the instance of \textit{glue}-(\( x=y \)) removed, if \( x, y \in \mathcal{V}_{\Gamma_j} \), or otherwise \( \Pi_j \).
Proposition 3.10.3. Every OCMT derivable in system OptSet\textsubscript{M} is also derivable in system OptSet\textsubscript{I}.

Proof. Consider a proof tree in system OptSet\textsubscript{M}. Then, without loss of generality, it has the structure described in lemma 3.10.2 on page 42. Applying the rewriting steps of proposition 3.10.1 on page 41 in reverse direction yields a proof tree in systems Opt\textsubscript{I} and OptSet\textsubscript{I} that derives the same OCMT. □

Theorem 3.10.4. System OptSet\textsubscript{I} derives opetopic sets, in the sense of theorems 3.6.15 and 3.6.17 on page 35 and on page 36.

Proof. By propositions 3.10.1 and 3.10.3 on page 41 and on the current page, the OCMTs derived by system OptSet\textsubscript{I} and OptSet\textsubscript{I} are the same. □

3.11. Examples

In this section, we give example derivations in system OptSet\textsubscript{I}. For clarity, we do not repeat the type of previously typed variables in proof trees.

Example 3.11.1. The opetopic set

\[
\begin{array}{c}
\circ \circ \circ \\
\circ \circ \circ \\
\circ \circ \circ
\end{array}
\]

\[
g \downarrow \alpha
\]

of example 3.7.1 on page 36 can be derived as follows. The first half of the proof tree is on the left, and the second half on the right. Moreover, for clarity, we do not repeat the typing of previously typed variables.
where rules of the form \((x = y)\) are shorthands for \(\text{glue}-(x = y)\).

**Example 3.11.2.** The opetopic set

![Diagram]

of example 3.7.2 on page 37 can be derived as follows.

\[
\begin{array}{c}
\vdash a : \emptyset \quad \vdash b : \emptyset \\
\vdash a, b \quad \text{point} \\
\vdash a, b + a : \emptyset \quad \text{sum} \\
\vdash a, b, t f : \emptyset \\
\vdash f : a \bullet b \\
\vdash b = t f \quad \text{shift} \\
\vdash b = t f \quad \text{glue}-(b = t f) \\
\vdash b = t f \quad \text{pd} \\
\vdash b = t f \quad \rightarrow_0 t f : \emptyset \\
\vdash b = t f \quad \text{shift} \\
\vdash b = t f \quad \text{pd} \\
\vdash b = t f \quad \rightarrow_0 a : \emptyset \\
\vdash b = t f \quad \text{shift} \\
\vdash b = t f \quad \text{pd} \\
\vdash b = t f \quad \rightarrow_1 g : t f \bullet \emptyset \\
\vdash b = t f \quad \text{graft} \\
\vdash b = t f \quad \rightarrow_1 g(t f \leftarrow f) : a \bullet \emptyset \\
\vdash b = t f \quad \text{shift} \\
\vdash b = t f \quad \text{glue}-(h = t a) \\
\vdash b = t f \quad \text{glue}-(a = t g) \\
\vdash b = t f \quad \text{glue}-(a = t h) \\
\vdash b = t f \quad \text{pd} \\
\vdash b = t f \quad \rightarrow_1 h : a \bullet \emptyset \\
\vdash b = t f \quad \text{shift} \\
\vdash a, b, t f, t g, t h, t t a : \emptyset \\
\vdash a, b, t f, t g, t h, t t a : \emptyset \\
\end{array}
\]
3.12. Python implementation

The system $\text{OptSet}^n_m$ is implemented in module `opetopy.NamedOpetopicSetM` of [Ho Thanh, 2018b]. Its usage is very similar to that of `opetopy.NamedOpetope` and `opetopy.NamedOpetopicSet`, presented in sections 3.4 and 3.8 on page 28 and on page 38 respectively.

Figure 3.12.1. Derivation of example example 3.7.1 on page 36 using `opetopy.NamedOpetopicSetM`

```python
from NamedOpetopicSetM import Glue, Pd, Point, RuleInstance, Shift, Sum

# We first derive the $f$, $g$ components
p1 = Shift(Pd(Point("a"), "a"), "f")
p1 = Shift(Pd(p1, "a"), "g")

# We then derive the $h$ component
p2 = Shift(Pd(Point("b"), "b"), "h")

# We proceed to sum the two, glue some cells, and introduce alpha
example = Sum(p1, p2)  # type: RuleInstance
type: RuleInstance
type: RuleInstance
type: RuleInstance
type: RuleInstance
type: RuleInstance
example = Glue(example, "b", "tf")
example = Glue(example, "b", "tg")
example = Shift(Pd(example, "f"), "alpha")
example = Glue(example, "b", "ttalpha")
example = Glue(example, "g", "talpha")
example = Glue(example, "a", "th")
```
CHAPTER 4

Unnamed approach

4.1. THE SYSTEM FOR OPETOPES

The unnamed approach for opetopes relies on the calculus of higher addresses presented in section 2.2.3 on page 11 to identify cells, rather than names as in the named approach. For example, recall the opetope \(3 \in \mathcal{O}_2\) from example 2.2.1 on page 12, drawn on the left, with its underlying \(3^0\)-tree represented on the right:

\[
\begin{array}{c}
\Rightarrow \\
\end{array}
\]

In the unnamed approach for opetopes presented in this section, \(3\) will be encoded as a mapping from its set of node addresses \(3^* = \{[], [\ast], [**]\}\) to the set of 1-opetopes \(\mathcal{O}_1\) as follows:

\[
3 \mapsto \begin{cases} 
[\ast] & \mapsto \diamond \\
[**] & \mapsto \diamond \\
\end{cases}
\]

The 1-opetope \(\diamond\) can recursively be encoded by \(\{\ast \mapsto \blacklozenge\) , which give a complete expression of \(3\):

\[
\begin{array}{c}
\overbrace{[\ast]} \mapsto \{ \blacklozenge \} \\
\end{array}
\]

This example will be treated in depth in example 4.3.2 on page 55. In a similar manner, the opetope \(\omega\) on the left, whose tree is given in the middle, can be encoded as on the right:

\[
\begin{array}{c}
\Rightarrow \\
\end{array}
\]

Similarly to \(3\), the opetope \(2\) can be expressed by \(\{[\ast] \mapsto \diamond\) , and recall that \(\diamond\) can be expressed by \(\{\ast \mapsto \blacklozenge\) . We thus have a complete encoding of \(\omega\) as:

\[
\begin{array}{c}
\overbrace{[\ast]} \mapsto \{ \blacklozenge \} \\
\end{array}
\]

47
This example is fully treated in example 4.3.3 on page 55.

4.1.1. **Preopetopes.** Now we define the set $\mathbb{P}_n$ of $n$-preopetopes by the following grammar:

\[
\begin{align*}
\mathbb{P}_- & ::= \emptyset & \text{by convention} \\
\mathbb{P}_0 & ::= \bullet \\
\mathbb{P}_n & ::= \begin{cases} 
\mathbb{A}_{n-1} \leftarrow \mathbb{P}_{n-1} \\
\vdots \\
\mathbb{A}_{n-1} \leftarrow \mathbb{P}_{n-1} \\
\mathbb{P}_{n-2} \end{cases} & n \geq 1 \\
\end{align*}
\]

(4.1.1)

\[
\begin{align*}
\mathbb{P}_n & ::= \begin{cases} 
\mathbb{A}_{n-1} \leftarrow \mathbb{P}_{n-1} \\
\vdots \\
\mathbb{A}_{n-1} \leftarrow \mathbb{P}_{n-1} \\
\mathbb{P}_{n-2} \end{cases} & n \geq 2
\end{align*}
\]

(4.1.2)

where the set $\mathbb{A}_n$ of $n$-addresses is defined in section 2.2.3 on page 11. In line (4.1.1), we require further that there is at least one $(n-1)$-address, and that all addresses are distinct. Further, a preopetope of this form is considered as a set of expressions $\mathbb{A}_{n-1} \leftarrow \mathbb{P}_{n-1}$ rather than a list: for instance, the following two $(n+1)$-preopetopes are equal

\[
\begin{align*}
\{[p_1] \leftarrow p_1, [p_2] \leftarrow p_2\} & = \{[p_2] \leftarrow p_2, [p_1] \leftarrow p_1\}
\end{align*}
\]

for any distinct $n$-addresses $[p_1], [p_2] \in \mathbb{A}_n$, and any $p_1, p_2 \in \mathbb{P}_n$. Here are examples of a 1, 2, 3, and 4-preopete respectively:

\[
\begin{align*}
\{\bullet \leftarrow \} & \leftarrow \{\bullet \leftarrow \} \leftarrow \{\bullet \leftarrow \} \\
\{\bullet \leftarrow \} & \leftarrow \{\bullet \leftarrow \} \leftarrow \{\bullet \leftarrow \} \\
\{\bullet \leftarrow \} & \leftarrow \{\bullet \leftarrow \} \leftarrow \{\bullet \leftarrow \} \\
\{\bullet \leftarrow \} & \leftarrow \{\bullet \leftarrow \} \leftarrow \{\bullet \leftarrow \}
\end{align*}
\]

As mentioned in the introduction of this chapter, the first corresponds to the unique 1-opetope $\bullet$. We will see that the second does not correspond to an actual opetope, as it is impossible for a 2-opetope to only contain addresses $\{\}$ and $\{\bullet \leftarrow \} \leftarrow \{\bullet \leftarrow \} \leftarrow \{\bullet \leftarrow \} \leftarrow \{\bullet \leftarrow \}$ (it would at least need addresses $\{\bullet \leftarrow \}$, $\{\bullet \leftarrow \}$, $\{\bullet \leftarrow \}$, and $\{\bullet \leftarrow \}$), and neither does the third, as it does not have a root node (corresponding to address $\{\}$). Finally, the last preopetope does represent an opetope.

An $n$-preopetope $p$ is **degenerate** if it is of the form of line (4.1.2), it is **non-degenerate** otherwise; we write $\dim p = n$ for its **dimension**. There is a unique 1-preopetope $\{\bullet \leftarrow \}$, which we write $\bullet$. If we have a non-degenerate $n$-preopetope of the form

\[
\begin{align*}
p = \begin{cases} 
[p_1] \leftarrow q_1 \\
\vdots \\
[p_k] \leftarrow q_k
\end{cases}
\end{align*}
\]

(4.1.3)

we call $[p_1], \ldots, [p_k]$ the **source addresses** of $p$ (or just **sources**), we write $p^{ullet}:=[[p_1], \ldots, [p_k]] \subseteq A_{n-1}$ the set of addresses of $p$, and $s(p):=q_1$, the $[p_1]$-**source** of $p$. Assume $n \geq 2$. A **leaf** of $p$ is an $(n-1)$-address of the form $[p[q]]$ such that $[p] \in p^{ullet}$, $[q] \in (s(p)p)^{ullet}$, and $[p[q]] \neq p^{ullet}$. We write $p^l \subseteq A_{n-1}$ for the set of leaves of $p$. By convention, if $p$ is degenerate, we set $p^\bullet:=\emptyset$ and $p^l:=\{\}$, and furthermore, $p^\bullet = p^l := \emptyset$.

**Definition 4.1.4** (Improper grafting). Let $p$ be an $n$-preopetope as in equation (4.1.3), and $q$ be an $(n-1)$-preopetope. If $[r] \in p^l$ is a leaf address of $p$, write

\[
\begin{align*}
p \stackrel{\circ}{\leftarrow} [r] q & := \\
\begin{cases} 
[p_1] \leftarrow q_1 \\
\vdots \\
[p_k] \leftarrow q_k \\
[r] \leftarrow q
\end{cases}
\end{align*}
\]

and call $p \circ [r] q$ the improper grafting of $q$ on $p$ at address $[r]$. By convention, the grafting operation is associative on the right.

For example,

\[
\begin{align*}
\begin{cases} 
\{\} \leftarrow \{\} \leftarrow \{\bullet \leftarrow \} \\
\{\} \leftarrow \{\bullet \leftarrow \} \leftarrow \{\bullet \leftarrow \} \\
\{\} \leftarrow \{\bullet \leftarrow \} \leftarrow \{\bullet \leftarrow \}
\end{cases}
& = \\
\begin{cases} 
\{\} \leftarrow \{\} \leftarrow \{\bullet \leftarrow \} \leftarrow \{\bullet \leftarrow \} \\
\{\} \leftarrow \{\} \leftarrow \{\bullet \leftarrow \} \leftarrow \{\bullet \leftarrow \} \leftarrow \{\bullet \leftarrow \}
\end{cases}
\end{align*}
\]
which, together with the introduction of this chapter, means that graphically,

\[
\begin{array}{c}
\begin{array}{c}
\text{original preopetope} \\
\end{array}
\end{array} \Rightarrow \begin{array}{c}
\begin{array}{c}
\text{new preopetope} \\
\end{array}
\end{array}
\]

The denomination “improper” is motivated by the fact that \( p \) and \( q \) do not have the same dimension. A definition of “proper” grafting is given for polynomial trees in section 2.1 on page 9. Any preopetope can be obtained by iterated improper grafting as follows.

**Lemma 4.1.5.** Let \( p \) be an \( n \)-preopetope as in equation (4.1.3) on the preceding page, and assume that whenever \( 1 \leq i < j \leq k \), we have either \([p_i] = [p_j]\), or \([p_i] \) and \([p_j] \) are \( \preceq \)-incomparable (in particular, this condition is satisfied if the \([p_i] \)’s are lexicographically sorted). Then

\[
p := \left( \cdots \left( \left[ p_1 \right] \leftarrow q_1 \right) \delta \left[ p_2 \right] q_2 \cdots \right) \delta \left[ p_k \right] q_k.
\]

*Proof.* The condition on the sequence \([p_1], \ldots, [p_k]\) guarantees that the successive improper graftings are well-defined. \( \square \)

### 4.1.2. Inference rules

We now introduce a typing system for preopetopes in order to characterize those corresponding to opetopes, which is formally shown in theorem 4.2.4 on page 54. We will deal with sequents of the form

\[
\Gamma \vdash p \rightarrow t,
\]

where \( p \) is an \( n \)-preopetope, \( t \) is an \((n-1)\)-preopetope, and the context \( \Gamma \) is a finite set of pairs consisting of addresses \([l] \in p\) and \([q] \in t^\ast\), denoted by \([l] [q] \). The preopetope \( p \) is the real object of interest, and as we will see in subsequent results, we may think of \( t \) as the “target” of \( p \), while \( \Gamma \) establishes a bijection between the leaves of \( p \) and the nodes of its target.

The operation of substitution, which consists in replacing a node by a pasting scheme in an opetope, can be defined as follows in our formalism.

**Definition 4.1.6 (Substitution).** Let \( t \) be a non degenerate \( n \)-preopetope, and \( \Gamma \vdash q \rightarrow u \) be a sequent, where \( q \) is an \( n \)-preopetope as well. Write \( t \) as

\[
t = \begin{cases} 
[t_1] \leftarrow w_1 \\
\vdots \\
[t_l] \leftarrow w_l 
\end{cases}
\]

For \([t_i] \in t^\ast\), we define \( t_{\sigma[t_i]} q \), the *substitution* by \( q \) in \( t \) at \([t_i] \), as follows:

1. If \( l = 1 \) and \( q \) is degenerate, then \( t_{\sigma[t_i]} q = q \);
2. If \( l \geq 2 \) and \( q \) is degenerate, then

\[
t_{\sigma[t_i]} q := \begin{cases} 
\chi[t_1] \leftarrow w_1 \\
\vdots \\
\chi[t_{i-1}] \leftarrow w_{i-1} \\
\chi[t_{i+1}] \leftarrow w_{i+1} \\
\vdots \\
\chi[t_l] \leftarrow w_l 
\end{cases}
\]

where \( \chi[t_j] := \begin{cases} 
[t_i] \cdots \\
[t_j] 
\end{cases} \)

(3) If \( i \geq 2 \), and \( q \) is not degenerate, write it as

\[
q = \begin{cases} 
[q_1] \leftarrow v_1 \\
\vdots \\
[q_k] \leftarrow v_k 
\end{cases}
\]
and define

\[ t \circ q := \begin{cases} 
\chi[t_i] \leftarrow w_i \\
\vdots \\
\chi[t_{i-1}] \leftarrow w_{i-1} \\
[t_i q_1] \leftarrow v_1 \\
[t_i q_k] \leftarrow v_k \\
\chi[t_{i+1}] \leftarrow w_{i+1} \\
\vdots \\
\chi[t_1] \leftarrow w_t 
\end{cases} \]

where \( \chi[t_j] := \begin{cases} [t_i a \ldots] & \text{if } [t_j] = [t_i b \ldots] \\
[t_j] & \text{otherwise.} \end{cases} \)

This operation relies on the context \( \Upsilon \), which we leave implicit. By convention, \( \circ \) is associative on the left.

Refer to section 4.3 on page 54 for examples of applications of this construction. We now state the inference rules of our unnamed system \( \text{Opt}^7 \) in figure 4.1.1.

**Figure 4.1.1. The \( \text{Opt}^7 \) system.**

**Introduction of points:**

\[ \vdash \bullet \rightarrow \varnothing \text{ point} \]

**Introduction of degeneracies:**

\[ \Gamma \vdash p \rightarrow t \quad \text{degen} \]

If \( \dim p = n \), remark that \( \dim \{ \{p\} \} = n + 2 \), and that \( \dim \{[\{p\} \leftarrow p] = n + 1 \).

**Shift to the next dimension:** Assume \( p^* = \{[p_1], \ldots, [p_k]\} \).

\[ \Gamma, [p_1] \vdash p \rightarrow t \quad \text{shift} \]

As in the previous rule, if \( \dim p = n \), then \( \dim \{[\{p\} \leftarrow p] = n + 1 \).

**Grafting:** Assume \( \dim p = n \geq 2 \), \( \dim q = n - 1 \), \( q^* = \{[s_1], \ldots, [s_l]\} \), and \( s[q] s[p] p = u \).

\[ \Gamma', [p[q][s_1]] \vdash p \rightarrow t \quad \Upsilon \vdash q \rightarrow u \quad \text{graft} \]

where \( \Gamma' \) is given by pairs of the form

1. \( \frac{a}{r[x]} \), where \( \frac{[a]}{[r[y][r']]} \in \Gamma \) and \( \frac{x}{y} \in \Upsilon \),

2. \( \frac{a}{b} \), where \( \frac{a}{b} \in \Gamma \) is not as above (i.e. \([b]\) not of the form \([r[y]][r']\) for some \( \frac{x}{y} \in \Upsilon \).

In large derivation trees, we will sometimes refer to this rule as \( \text{graft}=[p[q]] \) for clarity, or simply as \( [p[q]] \) in order to make notations lighter.

**Remark 4.1.7.** The transformation of context defined in rule \( \text{graft} \) in figure 4.1.1 might look complicated and unintuitive. And indeed it is, but we try to explain its purpose. Let \( \Gamma \vdash p \rightarrow t \) be a derivable sequent in \( \text{Opt}^7 \), with \( p \in \mathbb{P}_n \). As proved in lemma 4.1.9 on the facing page, \( \Gamma \) exhibits a bijection between \( p^* \) and \( t^* \). In theorem 4.2.4 on page 54, we prove that \( p \) corresponds uniquely to an \( n \)-opetope \( \omega = [p]^\text{poly} \), that \( t \omega = [t]^\text{poly} \), and that \( \Gamma \) corresponds to the readdressing function \( \rho_\omega : \omega^1 \rightarrow (t \omega)^* \) (see section 2.1.1 and appendix A.1 on page 10 and on page 71).
4.1. THE SYSTEM FOR OPETOPES

We now turn our attention to the named approach of chapter 3 on page 15. Applying theorem 3.2.22 on page 25, we know that ω corresponds to a unique sequent (modulo variable renaming)

\[ E \vdash \exists_n x : s x \mapsto s^2 x \mapsto \cdots \mapsto \varnothing \]

where \( x \in V \). More precisely, considered as a tree, ω is encoded by the term \( s x \), and by proposition 3.2.11 on page 22, \( t \omega \) is encoded by \( s^2 x \). In lemma 3.2.10 on page 22, we show that \( p \omega \) exhibits a bijection

\[ \{ \&_{s x} b \mid b \in V_{n-2}, b \in (s^2 x)^\star \} \xrightarrow{p} \{ \&_{s x} b \mid b \in V_{n-2}, b \in (s^2 x)^\star \} . \]

Say that a node of the term \( s^2 x \) is a variable \( b \in (s^2 x)^\star \), while a leaf of \( s x \) is a variable that can be used for grafting (see rule graft in figure 3.1.1 on page 17), i.e. a variable \( b \in (s^2 x)^\star \). Then left hand side can be considered as the set of leaf addresses of \( s x \), while the right hand side is its set of node addresses of \( s^2 x \), and \( p \omega \) maps the address of \( b \in V_{n-2} \) as a leaf of \( s x \) to the address of \( b \) as a node of \( s^2 x \). But here, the function \( p \omega \) is unnecessary: this correspondence is already established by the name of the variables!

In \( \text{Opt}^\circ \) however, such bookkeeping is paramount since there is no such things as names. Based on the previous discussion, we conclude that \( \Gamma \) is precisely the desired correspondence.

We now prove basic properties of derivable sequents in \( \text{Opt}^\circ \). In proof trees, we may sometimes omit irrelevant information: for instance, if contexts and targets are not important, the shift rule may be written as

\[
\frac{p}{\{[] \leftarrow p} \quad \text{shift}
\]

Lemma 4.1.8. If \( \Gamma \vdash p \longrightarrow t \) is a derivable sequent, then \( \dim p = \dim t + 1 \).

Proof. We proceed by induction. The only non trivial case is graft. Remark that \( \dim(p \circ_{[p[q]]} q) = \dim p = \dim q + 1 = \dim(t \circ [t]) q + 1 \).

Lemma 4.1.9. Let \( \Gamma \vdash p \longrightarrow t \) be a derivable sequent with \( \dim p \geq 2 \). Then \( \Gamma \) establishes a bijection between \( p^1 \) and \( t^* \) (i.e. as a set of pairs, \( \Gamma \) is the graph of a bijective function).

Proof. The fact that \( \Gamma \) is a relation from \( p \) to \( t^* \) (i.e. that whenever \( [a] \in \Gamma \) we have \( [a] \in p \) and \( [b] \in t^* \)) is clear from the inference rules. It is also clear that \( \Gamma \) is a function (i.e. that whenever \( [a], [a'] \in \Gamma \) we have \( [a] \neq [a'] \)). Finally, the fact that it is a bijection is clear in the case of degen and shift, and is a routine verification in the case of graft.

Proposition 4.1.10. If \( \Gamma \vdash p \longrightarrow t \) is derivable, then \( t \) is too.

Proof. The only non obvious case is (as always) graft, where we have to show that \( t \circ [t] q \) is derivable. Since the sequent \( \Gamma, [p[q]] \vdash p \longrightarrow t \) has a non empty context, \( p \) and \( t \) are non-degenerate. Write \( t \) and \( q \) as in definition 4.1.6 on page 49. Up to reindexing, assume that \( \chi[t_j] = [t_j] \) if and only if \( j < i \). Assume moreover that the sequence \( [t_1], \ldots, [t_{i-1}] \) is lexicographically ordered, and likewise for \( [t_{i+1}], \ldots, [t_l] \). For \( j > i \) write \( [t_j] = [t_i] [b_j] [x_j] \) and \( p[t_j] = [t_i] [a_j] [x_j] \), so that \( \Upsilon = \{ [a_j] \}_{i < j \leq l} \). Then the proof tree of \( t \circ [t] q \) is sketched as follows:

1. If \( [t_i] = [i] \), then necessarily \( i = 1 \), and \( t \circ [t] q \) is derived as follows:

\[
\begin{array}{c}
q \quad \vdots \\
\hline
q \circ_{[a_2 x_2]} v_2 \quad \text{graft-} [a_2 x_2] \quad \vdots \\
\hline
(q \circ_{[a_2 x_2]} v_2) \circ_{[a_3 x_3]} v_3 \quad \text{graft-} [a_3 x_3] \\
\hline
\vdots \\
\hline
(\cdots (q \circ_{[a_2 x_2]} v_2) \cdots) \circ_{[a_{l-1} x_{l-1}]} v_{l-1} \quad \vdots \\
\hline
(\cdots (q \circ_{[a_2 x_2]} v_2) \cdots) \circ_{[a_l x_l]} v_l \quad \text{graft-} [a_l x_l] \\
\end{array}
\]

and by definition, \((\cdots (q \circ_{[a_2 x_2]} v_2) \cdots) \circ_{[a_l x_l]} v_l = t \circ [t] q \).
Remark 4.1.12. Our syntax is closely related to the one given for multitopes [Hermida et al., 2002, Section 3], called here HMP. Briefly, in HMP, the unique 0 and 1-opetopes are respectively denoted \( \star \) and \# and, given an \( n \)-opetope \( p \), the notation \( \lceil p \rceil \) (resp. \( \lceil p^* \rceil \)) is used for the corresponding degenerated (resp. shifted) \((n+2)\) (resp. \((n+1)\)) opetope. The nodes of an opetope come equipped with a canonical order (just as in our system we could require opetopes to be always sorted according to the lexicographical order \( z \)), which apparently dispenses from using addresses. In HMP, an inductive definition of opetopes is given, in the same spirit as our sequent calculus: in particular, typing conditions involving targets when grafting opetopes (grafting is simply application in HMP) are involved. However, no explicit definition at the level of the syntax is given for computing targets (the description given resorts to multicategorical composition), and it is not clear to us how to define it without considering addresses and maintaining more information, as we do with our sequent calculus.

4.2. Equivalence with polynomial opetopes

We now show theorem 4.2.4 on page 54, stating that the elements of \( \mathbb{P}_n^\gamma \) are in bijective correspondence with the set \( \mathcal{O}_n \) of polynomial \( n \)-opetopes. To this end, we now construct a bijection \( \lceil - \rceil^\gamma : \mathcal{O}_n \to \mathbb{P}_n^\gamma \). If \( n = 0, 1 \), then both sets are singletons, so that \( \lceil - \rceil^\gamma \) is trivially defined. Assume \( n \geq 2 \), and that \( \lceil - \rceil^k \) is defined for \( k < n \). Recall that \( \mathcal{O}_n = \text{tr} \mathfrak{X}^{n-2} \). We distinguish three cases:
We prove that the last case makes sense in the following three lemmas.

**Proof.**

(1) (Degenerate case) for \( \phi \in \mathbb{O}_{n-2} \), let \( \llbracket 1_\phi \rrbracket^2 := \llbracket \llbracket \phi \rrbracket \rrbracket^2 \);  

(2) (Corollary case) for \( \psi \in \mathbb{O}_{n-1} \), let \( \llbracket Y_\psi \rrbracket^2 := \llbracket \llbracket \psi \rrbracket \rrbracket^2 \);  

(3) (Graft case) for \( \omega \in \mathbb{O}_n \) that is neither degenerate nor a corolla, write \( \omega = \nu \circ \llbracket l \rrbracket Y_\psi \), for \( \nu \in \mathbb{O}_n \), \( \psi \in \mathbb{O}_{n-1} \), and \( \llbracket l \rrbracket \in \nu^l \), then let \( \llbracket \omega \rrbracket^2 \) be defined by the following proof tree:

\[
\frac{\llbracket \nu \rrbracket^2 \llbracket \psi \rrbracket^2}{\llbracket \omega \rrbracket^2} \xrightarrow{\text{graft-} \llbracket l \rrbracket} \]

or in other words, \( \llbracket \omega \rrbracket^2 = \llbracket \nu \rrbracket^2 \circ_{\llbracket l \rrbracket} \llbracket \psi \rrbracket^2 \).

We prove that the last case makes sense in the following three lemmas.

**Lemma 4.2.1.** For \( \omega \in \mathbb{O}_n \), we have \( \omega^* = \llbracket \omega \rrbracket^* \), and if \( \omega \) is non degenerate, then \( \omega^l = \llbracket \omega \rrbracket^l \).

**Proof.** We proceed by opetopic induction (see remark 2.2.3 on page 12).

(1) (Degenerate case) Assume \( \omega = 1_\phi \), for some \( \phi \in \mathbb{O}_{n-2} \). Then \( \omega^* = \emptyset = \left( \llbracket \llbracket \phi \rrbracket \rrbracket^2 \right)^* \). For leaves: \( \omega^l = \emptyset = \left( \llbracket \llbracket \phi \rrbracket \rrbracket^2 \right)^l \).

(2) (Corollary case) Assume \( \omega = Y_\psi \), for some \( \psi \in \mathbb{O}_{n-1} \), so that we have \( \llbracket \omega \rrbracket^2 = \llbracket \llbracket \psi \rrbracket \rrbracket^2 \). Then \( \omega^* = \{[\ ]\} = \left( \llbracket \llbracket \psi \rrbracket \rrbracket^2 \right)^* \). By induction, \( \psi^* = \llbracket \psi \rrbracket^* \), so the leaf addresses of \( \omega \) and \( \{[\ ]\} \) are both of the form \( \{[\ ]\} \), hence \( \omega^l = \llbracket \omega \rrbracket^l \).

(3) (Graft case) Assume \( \omega = \nu \circ \llbracket l \rrbracket Y_\psi \), for some \( \nu \in \mathbb{O}_n \), \( \psi \in \mathbb{O}_{n-1} \), and \( \llbracket l \rrbracket \in \nu^l \). Then, by induction,

\[
\omega^* = \nu^* + \{[\ ]\} = \llbracket \nu \rrbracket^* + \{[\ ]\} = \left( \llbracket \nu \rrbracket^2 \circ_{\llbracket l \rrbracket} \llbracket \psi \rrbracket^2 \right)^* = \llbracket \omega \rrbracket^* .
\]

Similarly, by induction,

\[
\omega^l = \nu^l - \{[\ ]\} + \{[[q] \mid [q] \in \psi^* \}
\]

\[
= \llbracket \nu \rrbracket^l - \{[\ ]\} + \{[[q] \mid [q] \in \psi^* \}
\]

\[
= \left( \llbracket \nu \rrbracket^2 \circ_{\llbracket l \rrbracket} \llbracket \psi \rrbracket^2 \right)^l = \llbracket \omega \rrbracket^l .
\]

and we conclude. \( \square \)

**Lemma 4.2.2.** For \( \omega \in \mathbb{O}_n \) and \( [p] \in \omega^* \), we have \( \llbracket s_{[p]} \omega \rrbracket^2 = s_{[p]} \llbracket \omega \rrbracket^2 \).

**Proof.** We proceed by opetopic induction.

(1) (Degenerate case) If \( \omega \) is degenerate, then \( \omega^* = \emptyset \), so there is nothing to check in this case.

(2) (Corollary case) Assume \( \omega = Y_\psi \), for some \( \psi \in \mathbb{O}_{n-1} \). Then \( \omega^* = \{[\ ]\} \), and \( \llbracket s_{[\ ]} \omega \rrbracket^2 = \llbracket \psi \rrbracket^2 = s_{[\ ]} \llbracket \psi \rrbracket^2 = s_{[\ ]} \llbracket \omega \rrbracket^2 \).

(3) (Graft case) Assume \( \omega = \nu \circ \llbracket l \rrbracket Y_\psi \), for some \( \nu \in \mathbb{O}_n \), \( \psi \in \mathbb{O}_{n-1} \), and \( \llbracket l \rrbracket \in \nu^l \). Let \( [p] \in \omega^* \). If \( [p] = [\ ] \), then

\[
\llbracket s_{[\ ]} \omega \rrbracket^2 = \llbracket \psi \rrbracket^2 = s_{[\ ]} \left( \llbracket \nu \rrbracket^2 \circ_{\llbracket l \rrbracket} \llbracket \psi \rrbracket^2 \right) = s_{[\ ]} \llbracket \omega \rrbracket^2 .
\]

Otherwise, we have

\[
\llbracket s_{[p]} \omega \rrbracket^2 = \llbracket s_{[p]} \nu \rrbracket^2 = s_{[p]} \llbracket \nu \rrbracket^2 = s_{[p]} \left( \llbracket \nu \rrbracket^2 \circ_{\llbracket l \rrbracket} \llbracket \psi \rrbracket^2 \right) = s_{[p]} \llbracket \omega \rrbracket^2 .
\]

\( \square \)

**Lemma 4.2.3.** For \( \omega \in \mathbb{O}_n \), we have \( \llbracket t \omega \rrbracket^2 = t \llbracket \omega \rrbracket^2 \), and \( \varphi_{\omega} = \varphi_{\llbracket \omega \rrbracket^1} \).

**Proof.** We proceed by opetopic induction.
(1) (Degenerate case) Assume \( \omega = l_\phi \), for some \( \phi \in \Omega_{n-2} \). Then \( \llbracket t\omega \rrbracket^\gamma = \llbracket Y_\phi \rrbracket^\gamma = \{[\] \mapsto \phi\}^\gamma = t\{\{\phi\}\}^\gamma = t\llbracket \omega \rrbracket^\gamma \). Since \( \omega \) and \( \llbracket \omega \rrbracket^\gamma \) are both degenerate (as opetope and preopetope, respectively), \( \rho_\omega \) and \( \rho_{\llbracket \omega \rrbracket^\gamma} \) both map \([\] \) (the unique leaf of \( \omega \) and \( \llbracket \omega \rrbracket^\gamma \)) to \([\] \) (the unique node of \( t\omega \) and \( t\llbracket \omega \rrbracket^\gamma \)).

(2) (Corollary case) Assume \( \omega = Y_\psi \), for some \( \psi \in \Omega_{n-1} \), so that we have \( \llbracket \omega \rrbracket^\gamma = \{[\] \mapsto \psi\}^\gamma \). Then \( \llbracket t\omega \rrbracket^\gamma = \llbracket \psi \rrbracket^\gamma = t\{[\] \mapsto \psi\}^\gamma = t\llbracket \omega \rrbracket^\gamma \). Moreover, we have \( \omega^\dagger = \llbracket \omega \rrbracket^\gamma \), \( \{[\] \mapsto \psi\}^\gamma \), and by definition, \( \rho_\omega([p]) = [p] = \rho_{\llbracket \omega \rrbracket^\gamma}([p]) \).

(3) (Graft case) Assume \( \omega = \nu \circ [t] \cdot Y_\psi \), for some \( \nu \in \Omega_n \), \( \psi \in \Omega_{n-1} \), and \([t] \in \nu^\gamma \). Then,

\[
\llbracket t\omega \rrbracket^\gamma = \llbracket (t\nu) \circ \rho_{\nu}[t] \cdot \llbracket \psi \rrbracket^\gamma \rrbracket^\gamma
\]

by \( \text{ind. } \rho_\nu = \rho_{\nu} \).

Lastly, we prove \( \rho_\omega = \rho_{\llbracket \omega \rrbracket^\gamma} \). Let \([u] \in \omega \).

(a) If \([u] = [t[q]] \) for \([q] \in \psi^\gamma \), then \( \rho_\omega([q]) = \rho_{\nu}[t] \cdot [q] \) (see line (A.2.3)), where \( U = t\nu \), \( T = \psi \), and \([u] = \rho_{\nu}[t] \). On the other hand, \( \rho_{\llbracket \omega \rrbracket^\gamma}([q]) = \rho_{\llbracket \psi \rrbracket^\gamma}([t[q]]) \).

(b) If \([u] \in \nu^\gamma \) is of the form \( \rho_{\nu}[u] = [p\rho_{\nu}[v]]^\gamma \), for some \([v] \in \psi^\gamma \), then \( \rho_\omega[u] = [p]\rho_{\nu}[v] \). On the other hand, by induction, \( \rho_{\llbracket \nu \rrbracket^\gamma}([u]) = \rho_{\nu}[u] = [p\rho_{\nu}[v]]^\gamma \), and by definition we have \( \rho_{\llbracket \omega \rrbracket^\gamma}([u]) = [p]\rho_{\nu}[v] \).

(c) If \([u] \in \nu^\gamma \) is not of the form above, then \( \rho_\omega[u] = \rho_{\nu}[u] = \rho_{\llbracket \nu \rrbracket^\gamma}([u]) = \rho_{\llbracket \nu \rrbracket^\gamma}([u]) \).

We finally prove that \( \llbracket \cdot \rrbracket^\gamma \) is a bijection by constructing its inverse \( \llbracket \cdot \rrbracket^{\poly} : \mathbb{P}_n^\gamma \rightarrow \Omega_n \). If \( n = 0, 1 \), then both sets are singletons, so that \( \llbracket \cdot \rrbracket^{\poly} \) is trivially defined. Assume \( n \geq 2 \), and that \( \llbracket \cdot \rrbracket^{\poly} \) is defined for \( k < n \). We distinguish three cases:

1. (Degenerate case) if \( q \in \mathbb{P}_{n-2} \), then \( \llbracket q \rrbracket^{\poly} = \llbracket I_q \rrbracket^{\poly} \).
2. (Corollary case) if \( q \in \mathbb{P}_{n-1}^\gamma \), then \( \llbracket [q] \rrbracket^{\poly} = \llbracket I_q \rrbracket^{\poly} \).
3. (Graft case) for \( p \in \mathbb{P}_n^\gamma \), \( q \in \mathbb{P}_{n-1}^\gamma \), and \([t] \in \nu^\gamma \), define \( \llbracket p \circ [t] \cdot q \rrbracket^{\poly} = \llbracket p \rrbracket^{\poly} \circ \llbracket [t] \rrbracket^{\poly} \cdot \llbracket q \rrbracket^{\poly} \).

**Theorem 4.2.4.** The functions \( \llbracket \cdot \rrbracket^\gamma \) and \( \llbracket \cdot \rrbracket^{\poly} \) are mutually inverse. Moreover, for \( n \geq 2 \) and \( \omega \in \Omega_n \), we have

(1) \( \omega^\dagger = \llbracket \omega \rrbracket^\gamma \), and if \( \omega \) is non-degenerate, \( \omega^\gamma = \llbracket \omega \rrbracket^\gamma \).
(2) for \([p] \in \omega^\gamma \), \( \llbracket s[p] \omega \rrbracket^\gamma = s[p] \llbracket \omega \rrbracket^\gamma \), and \( \llbracket t\omega \rrbracket^\gamma = t\llbracket \omega \rrbracket^\gamma \).
(3) \( \rho_\omega = \rho_{\llbracket \omega \rrbracket^\gamma} \).

**4.3. Examples**

In this section, we give example derivations in system Opt⁷.

**Example 4.3.1** (The arrow). The unique 1-opetope \( \star = \{ \star \mapsto \bullet \} \) is derived by

\begin{align*}
&\text{point} & \rightarrow \star \mapsto \bullet \rightarrow \circ \mapsto \bullet \\
&\text{shift} & \rightarrow \{ \star \mapsto \bullet \}
\end{align*}
Example 4.3.2 (Opetopic integers). As in example 3.3.2 on page 26, we provide the derivation of opetopic integers \( n \in \mathbb{O}_2 \). The derivation of 0 is given on the left, whereas for \( n \geq 1 \), the opetope \( n \) is derived as on the right:

\[
\begin{align*}
&\vdash \bullet \rightarrow \flat \quad \text{shift} \quad \vdash \bullet \rightarrow \flat \\
\text{point} \quad \vdash \star \rightarrow \diamond \\
&\vdash \star \rightarrow \diamond \quad \text{degen}
\end{align*}
\]

\[
\begin{align*}
&\vdash \bullet \rightarrow \flat \\
&\vdash \star \rightarrow \diamond \\
&\vdash \star \rightarrow \diamond
\end{align*}
\]

where \( \star^n = \cdots \star \), and where there is a total of \( n - 1 \) instances of the \textit{graft} rule.

Example 4.3.3 (A classic). The 3-opetope on the left is derived as on the right:

\[
\begin{align*}
&\vdash \bullet \rightarrow \star \rightarrow \diamond \\
&\vdash \star \rightarrow \diamond
\end{align*}
\]

\[
\begin{align*}
&\vdash \bullet \rightarrow \star \rightarrow \diamond \\
&\vdash \star \rightarrow \diamond \\
&\vdash \star \rightarrow \diamond
\end{align*}
\]

Example 4.3.4 (A degenerate case). The 3-opetope on the left is derived as on the right:

\[
\begin{align*}
&\vdash \bullet \rightarrow \star \rightarrow \diamond \\
&\vdash \star \rightarrow \diamond
\end{align*}
\]

\[
\begin{align*}
&\vdash \bullet \rightarrow \star \rightarrow \diamond \\
&\vdash \star \rightarrow \diamond
\end{align*}
\]

Example 4.3.5 (Another degenerate case). The 3-opetope on the left is derived as on the right:

\[
\begin{align*}
&\vdash \bullet \rightarrow \star \rightarrow \diamond \\
&\vdash \star \rightarrow \diamond
\end{align*}
\]

Example 4.3.6 (A 4-opetope). The 4-opetope
is derived by

\[
\begin{align*}
\vdash [s^+] & \to 3 \to s \\
\vdash [s^+] & \to [s^+] \to 3 \to 3 \\
\vdash [s^+] & \to [s^+] \to 2 \to s \\
\vdash [s^+] & \to [s^+] \to 4 \to [s^+] \\
\vdash [s^+] & \to [s^+] \to 5 \to [s^+] \\
\vdash [s^+] & \to [s^+] \to 1 \to [s^+] \\
\end{align*}
\]

where rule $s$ is a shorthand for shift.

4.4. Deciding opetopes

We now present in algorithm 4.4.1 the isOpetope function that, given a preopetope $\omega \in \mathbb{P}$, decides if $\omega \in \mathbb{P}^r$, as proved in proposition 4.4.1 on the next page. This algorithm tries to deconstruct $\omega$ by finding the last rule instance if its potential proof tree, and recursively checking the validity of the premises. We emphasize that this algorithm, while straightforward, is extremely inefficient.

**Algorithm 4.4.1 Well formation algorithm**

1: **procedure** isOpetope($\omega \in \mathbb{P}$) \>

2: if $\omega = \ast$ then
3: return true
4: else if $\omega = \emptyset \phi$ then
5: return isOpetope($\phi$)
6: else
7: while $\omega$ has an address of the form $[p\,q]$ do
8: if $[p] \neq \ast$ or not isOpetope($s[p] \omega$) then
9: return false
10: else if $[q] \neq (s[p]) \ast$ or not isOpetope($s[p] \omega$) then
11: return false
12: else if $t s[p] \omega \neq s[p] \omega$ then
13: return false
14: end if
15: Remove address $[p\,q]$ from $\omega$
16: end while
17: if $\omega$ is of the form $[\{} \leftarrow \psi$ then
18: return isOpetope($\psi$
19: else
20: return false
21: end if
22: end if
23: end procedure
Proposition 4.4.1. For \( \omega \in \mathbb{P} \), the execution is\textsc{opetope}() returns \texttt{true} if and only if \( \omega \in \mathbb{P}^\vee \).

Proof. This algorithm tries to deconstruct the potential proof tree of \( \omega \) in system \textsc{opt}?:

1. condition at line (2) removes an instance of the \texttt{point} rule;
2. condition at line (4) removes an instance of \texttt{degen};
3. each iteration of the \texttt{while} loop at line (7) removes an instance of \texttt{graft};
4. finally the condition at line (18) removes an instance of \texttt{shift}.

If the algorithm encounters an expression that is not the conclusion of any instance of any rule of \textsc{opt}?, it returns \texttt{false}. Otherwise, if all branches of the proof tree lead to \( \bullet \), it returns \texttt{true}. \( \square \)

4.5. Python implementation

The derivation system \textsc{opt}? and all required syntactic constructs are implemented in module \texttt{opetopy.UnnamedOpetope} of [Ho Thanh, 2018b]. In particular, the four derivation rules are represented by functions of the same name: \texttt{point}, \texttt{degen}, \texttt{shift}, and \texttt{graft}. As an example, we review the implementation of \texttt{shift} in figure 4.5.1.

\textbf{Figure 4.5.1.} Implementation of \textsc{opt}?’s \texttt{shift} rule in \texttt{opetopy.UnnamedOpetope.shift}

```python
# This function takes a sequent (opetopy.UnnamedOpetope.Sequent) and returns a sequent. A sequent seq is structured as follows: for seq = \( \Gamma \vdash s \rightarrow t \) we have seq.context = \( \Gamma \), seq.source = \( s \), and seq.target = \( t \).
def shift(seq: Sequent) -> Sequent:
    # We let n be the dimension of the preopetope s = seq.source
    n = seq.source.dimension
    # We construct a new context ctx dimension n + 1
    ctx = Context(n + 1)
    # We let ctx = \{[a] | a \in s^*\}
    for a in seq.source.nodeAddresses():
        ctx += (a.shift(), a)
    # We return the sequent ctx\vdash \{[\bullet]\} \leftarrow s \rightarrow s
    return Sequent(
        ctx,
        Preopetope.fromDictOfPreopetopes({
            Address.epsilon(n): seq.source
        })),
        seq.source
    )
```

To construct proof trees, those rules are further abstracted in classes \texttt{Point}, \texttt{Degen}, \texttt{Shift}, as well as \texttt{Graft}. We review the implementation of some examples in figures 4.5.2 to 4.5.4 on pages 58–59.
Figure 4.5.2. Derivation of the arrow sequent (see example 4.3.1 on page 54) using opetopy.UnnamedOpetope

```python
from opetopy.UnnamedOpetope import *

# The arrow sequent is obtained by an application of the point rule followed by an application of the shift rule.
arrow = Shift(Point())
# Note that the function opetopy.UnnamedOpetope.Arrow can be used to concisely get the proof tree of ◦.
```

Figure 4.5.3. Derivation of some opetopic integers (see example 4.3.2 on page 55) using opetopy.UnnamedOpetope

```python
from opetopy.UnnamedOpetope import *

# The opetopic integer 0 is obtained by an instance of the point rule, followed by an application of the degen rule.
opetopic_integer_0 = Degen(Point())
# The opetopic integer 1 is obtained by applying rule shift to the arrow ◦ as defined in the previous figure.
opetopic_integer_1 = Shift(arrow)
# The opetopic integer 2 is defined by 2 ◦ = 1 ◦[●]. The address [●] is obtained with the convenient UnnamedOpetope.address function (as opposed to using the UnnamedOpetope.Address class).
opetopic_integer_2 = Graft(opetopic_integer_1, arrow, address(["●"])
# Likewise, 3 = 2 ◦[●●].
opetopic_integer_3 = Graft(opetopic_integer_2, arrow, address(["●", "●"])
# Note that the function opetopy.UnnamedOpetope.OpetopicInteger can be used to get the proof tree of an arbitrary opetopic integer.
```
4.6. THE SYSTEM FOR OPETOPIC SETS

We now present OptSet\(^\dagger\), a derivation system for opetopic set that is \textit{controlled} by Opt\(^\dagger\) (unlike system OptSet\(^\ddagger\) that is \textit{based} on Opt\(^\dagger\), see figures 3.1.1 and 3.5.1 on page 17 and on page 30).

As always, contexts are considered as sets, so that the order in which the typings are written is irrelevant, even though those typings might be interdependent. We rely on two types of judgment that can be understood as follows:

(1) \(\Gamma \vdash \text{context}\) means that \(\Gamma\) is a well-formed context.

(2) \(\Gamma \vdash P\) means that in context \(\Gamma\), \(P\) is a well-formed pasting diagram.

We now state the inference rules in figure 4.6.1 on the next page, and simultaneously assign a \textit{shape} \(x^\kappa\) to any variable \(x\) in a derivable context.
Proof.

Lemma 4.7.1. In the setting above, we have $\sigma_i^k = x_i^k$.

Proof. We proceed by induction. By definition, $\sigma_1 = x_1$, and thus $\sigma_1^k = x_1^k$. By induction, take $1 \leq i \leq k$ and assume $\sigma_j^k = x_j^k$ for $j < i$. Since the shape of a pasting diagram $P = \begin{cases} \{[p_1] \leftarrow y_1 \} \end{cases}$ only depends on the shape of the
Let \( \text{Ctx}^2 \) be the syntactic category of our sequent calculus, i.e. the category whose objects are derivable contexts, and morphisms are substitutions as defined above. We now construct the unnamed stratification functor \( S' : (\text{Ctx}^2)^{op} \rightarrow \text{Fin}\overline{\Omega} \). For \( \Gamma \in \text{Ctx}^2 \) and \( \omega \in \Omega \), let

\[
S^2\Gamma_\omega = \left\{ x \in \Gamma \mid x^h = \| \omega \|^2 \right\}.
\]

If \( x^h \neq \bullet \), then the type \( X \) of \( x \) is of the form \( P \rightarrow z \), and we let \( tx : z \). This is well defined as by construction of \( \Gamma \) we have \( z^3 = t x^3 \). For \( [p] \in \omega^* \), we let \( s_{[p]} x := s_{[p]} P \). Again, this is well defined as \( (s_{[p]} P)^h = s_{[p]} P^h = s_{[p]} x^h \).

From there, the opetopic identities clearly hold, and \( S^2 \Gamma \) is a finite opetopic set. To abbreviate notations, and for \( f : \psi \rightarrow \omega \) a morphism in \( \Omega \), we let \( f = S^2 \Gamma f : S^2\Gamma_\omega \rightarrow S^2\Gamma_\psi \).

Take now \( \sigma = (\sigma_1, \ldots, \sigma_k) : \Gamma \rightarrow \Gamma \) a substitution, where as before \( \Gamma = (x_1 : T_1, \ldots, x_k : T_k) \). We define a morphism \( S^2 \sigma : S^2 \Gamma \rightarrow S^2 \Upsilon \). For \( x_i \) a variable of \( \Gamma \), and \( \omega \in \Omega \) such that \( \| \omega \|^2 = x^3 \), there is a corresponding cell \( x_i \in S^2\Gamma_\omega \), and we let \( S^2 \sigma(x_i) := \sigma_i \). This is well-defined since by lemma lemma \ref{lem:ctx2-fin-cat} on the facing page, we have \( \sigma_i \in S^2\Gamma_\omega \).

**Lemma 4.7.2.** The mapping \( S^2 \sigma \) is a morphism of opetopic sets (i.e. a natural transformation) \( S^2\Gamma \rightarrow S^2 \Upsilon \).

**Proof.** Assume \( \omega \neq \bullet \), so that the type of \( x_i \) is \( P \rightarrow x_j \) for some \( j < i \), and the type of \( \sigma_i \) is \( P[\sigma_1/x_1] \ldots [\sigma_{i-1}/x_{i-1}] \rightarrow \sigma_j \). Then \( S^2 \sigma(t x_i) = \sigma_j = t(S^2 \sigma(x_i)) \). If \( [p] \in \omega^* \), then \( s_{[p]} x_i = x_j \), for some \( j \leq i \), and

\[
S^2 \sigma(s_{[p]} x_i) = \sigma_i = s_{[p]} (P[\sigma_1/x_1] \ldots [\sigma_{i-1}/x_{i-1}]) = s_{[p]} (S^2 \sigma(x_i)).
\]

Hence, \( S^2 \sigma \) is a morphism of opetopic sets \( S^2\Gamma \rightarrow S^2 \Upsilon \). \( \square \)

**Theorem 4.7.3.** The functor \( S^2 : (\text{Ctx}^2)^{op} \rightarrow \text{Fin}\overline{\Omega} \) is an equivalence of categories.

**Proof.** Let \( \Gamma, \Upsilon \in \text{Ctx}^2 \), with \( \Gamma = (x_1 : T_1, \ldots, x_k : T_k) \), and \( f : S^2 \Gamma \rightarrow S^2 \Upsilon \). Then \( f = S^2(f(x_1), \ldots, f(x_k)) \), showing that \( S^2 \) is fully faithful. We now prove that it is essentially surjective. Take \( X \in \text{Fin}\overline{\Omega} \), and enumerate its cells as \( x_1 \in X_{\omega_1}, \ldots, x_k \in X_{\omega_k} \), such that whenever \( i < j \) we have \( \dim \omega_i \leq \dim \omega_j \). To each \( 0 \leq i \leq k \) we associate a derivable context \( \Gamma^{(i)} = (x_1 : T_1, \ldots, x_i : T_i) \) such that \( \| \omega_i \|^2 = \| \omega_i \|^2 \), as follows:

1. If \( \omega_i = \bullet \), let \( \Gamma^{(i)} \) be given by the following proof tree (so that implicitly, \( T_i = \bullet \)):

   \[
   \frac{\Gamma^{(i-1)} \text{ context}}{\Gamma^{(i-1)}, x_i : \bullet \text{ context}}
   \]

2. Assume \( \omega_i \neq \bullet \) is not degenerate. Then, by induction, for \( x_j = t x_i \), we have \( t x_i = t \omega_i \), and for all address \( [p_j] \in \omega_i^* \), we have \( s_{[p_j]} x_i = s_{[p_j]} \| \omega_i \|^2 \). From this, \( \Gamma^{(i)} \) is given by the following proof tree:

   \[
   \frac{\Gamma^{(i-1)} \text{ context}}{\Gamma^{(i-1)}, x_i : [p_j] \leftarrow s_{[p_j]} x_i \text{ graft}}
   \]

3. Assume \( \omega_i \) is degenerate. Then \( \Gamma^{(i)} \) is given by the following proof tree:

   \[
   \frac{\Gamma^{(i-1)} \text{ context}}{\Gamma^{(i-1)}, x_i : \{ t t x_i \leftarrow t x_i \text{ shift} \}}
   \]

Finally, the mapping \( x_i \mapsto x_i \) exhibits an isomorphism \( X \rightarrow S^2 \Gamma^{(k)} \). \( \square \)

The category \( \text{Ctx}^2 \) has finite limits, induced from finite colimits in \( \text{Fin}\overline{\Omega} \) through \( S^2 \). We conclude this section with a result similar to theorem 5.6.17 on page 36, stating that opetopic sets essentially are “models of the algebraic theory \( \text{Ctx}^2 \)”.
Theorem 4.7.4. We have an equivalence $\hat{\mathcal{O}} \cong \text{Mod}(\text{Ctx}^2, \text{Set})$, where $\text{Mod}$ is defined in theorem 3.6.17 on page 36.

Proof. We proceed as in the proof of theorem 3.6.17 on page 36.

4.8. Examples

In this section, we give example derivations in system OptSet$^2$. For clarity, we do not repeat the type of previously typed variables in proof trees.

Example 4.8.1. We now derive the following opetopic set, which is not representable:

\[ \begin{array}{c}
\downarrow \\
g \\
\bullet \downarrow \\
\circ \\
\downarrow \\
h \\
\end{array} \]

We decide to introduce all points first:

$\text{point} a : \bullet \quad \text{context} \quad a \Downarrow \quad \text{point} \quad a, b : \bullet \quad \text{context}$

Then we introduce $f$, by first specifying its source pasting diagram with the graft rule, parameterized by opetope $\bullet = \{ * \leftarrow \bullet \}$, and then applying the shift rule:

\[ \begin{array}{c}
\text{graft} \quad a, b \quad \text{context} \\
\UpwardParbox{a, b \{ * \leftarrow a \}} \\
\text{shift} \quad a, b, f : \{ * \leftarrow a \rightarrow b \quad \text{context} \}
\end{array} \]

We proceed similarly for $g$ and $h$:

\[ \begin{array}{c}
\text{graft} \quad a, b, f \quad \text{context} \\
\UpwardParbox{a, b, f \{ * \leftarrow a \}} \\
\text{shift} \quad a, b, f, g : \{ * \leftarrow a \rightarrow b \quad \text{context} \}
\end{array} \]

\[ \begin{array}{c}
\text{graft} \quad a, b, f, g \quad \text{context} \\
\UpwardParbox{a, b, f, g \{ * \leftarrow b \}} \\
\text{shift} \quad a, b, f, g, h : \{ * \leftarrow b \rightarrow a \quad \text{context} \}
\end{array} \]

Lastly, we introduce $\alpha$, first by specifying its source with the graft rule, parameterized by opetope $1 = \{ [] \leftarrow \bullet \}$ (see the opetopic integers defined in example 4.3.2 on page 55), and applying the shift rule:

\[ \begin{array}{c}
\text{graft} \quad a, b, f, g, h \quad \text{context} \\
\UpwardParbox{a, b, f, g, h \{ [] \leftarrow f \}} \\
\text{shift} \quad a, b, f, g, h, \alpha : \{ [] \leftarrow f \rightarrow g \quad \text{context} \}
\end{array} \]

Example 4.8.2 (A classic, maximally folded). We derive the following opetopic set

\[ \begin{array}{c}
\text{graft} \quad a, b, f, g, h, \alpha : \{ [] \leftarrow f \rightarrow g \quad \text{context} \}
\end{array} \]

where all 0-cells are the same cell $a$, all 1-cells are $f$, the 2-cells on the left are $\alpha$, the 2-cell on the right is $\beta$, and the 3-cell is $A$. Note that those identifications make the opetopic set not representable. We first derive $a$ and $f$:

$\text{point} a : \bullet \quad \text{context} \quad a \Downarrow \quad \text{graft} \quad a \rightarrow \{ * \leftarrow a \}$

$\text{shift} \quad a, f : \{ * \leftarrow a \rightarrow a \quad \text{context} \}$
Then we introduce $\alpha$, by first specifying its source pasting diagram with the \texttt{graft} rule, parameterized by opetope $2 = \{[] \leftarrow *\}$ (see the opetopic integers defined in example 4.3.2 on page 55), and applying the \texttt{shift} rule:

\[
\vdash a, f : \text{context} \quad \texttt{graft} \quad a, f : \begin{cases} [] & \leftarrow f \\ ![\star] & \leftarrow f \end{cases} \quad \texttt{shift} \quad a, f, \alpha : \begin{cases} [] & \leftarrow f \\ ![\star] & \leftarrow f \rightarrow f \end{cases} \\
\]

Likewise, we introduce $\beta$, where the \texttt{graft} rule is parameterized by $3$:

\[
\vdash a, f, \alpha, \beta : \begin{cases} [] & \leftarrow f \\ ![\star] & \leftarrow f \rightarrow f \\ ![\star\star] & \leftarrow f \end{cases} \\
\]

Lastly, we introduce $A$, where the \texttt{graft} rule is parameterized by $\begin{cases} [] & \leftarrow 2 \\ ![\star] & \leftarrow 2 \\ ![\star\star] & \leftarrow 2 \end{cases}$:

\[
\vdash a, f, \alpha, \beta, A : \begin{cases} [] & \leftarrow \alpha \\ ![\star] & \leftarrow \alpha \rightarrow \beta \end{cases} \\
\]

\textbf{Example 4.8.3.} The opetopic set

\[
\begin{array}{ccc}
& b & \downarrow_{\alpha} \\
f & & h \\
\end{array}
\]

is straightforwardly derived as
4. UNNAMED APPROACH

4.9. Python implementation

System OPTSET is implemented in module opetopy.UnnamedOpetopicSet of [Ho Thanh, 2018b]. The rules are represented by functions point, degen, graft, and shift, and are further encapsulated in rule instance classes Point, Degen, Graft, and shift.
Figure 4.9.1. Derivation of example 4.8.2 on page 62 using \texttt{opetopy.NamedOpetope}

```python
from opetopy.UnnamedOpetopicSet import *
from opetopy.UnnamedOpetope import address, Arrow, OpetopicInteger
from opetopy.UnnamedOpetope import Graft as OptGraft
from opetopy.UnnamedOpetope import Shift as OptShift

# We first derive the unique point $a$.
classic = Point("a")

# We then derive $f$ by firstly specifying a pasting diagram with the graft rule. It is
# constructed with a proof tree of its shape opetope (in system OPT'), and an
# address-to-variable mapping.

classic = Graft(pastingDiagram(
  Arrow(),
  {
    address([], 0): "a" # $a \mapsto a$
  }, classic)

# We then derive $f$.
classic = Fill("a", "f", classic)

# In a similar way, we derive $\alpha$ of shape 2.
classic = Graft(pastingDiagram(
  OpetopicInteger(2),
  {
    address([], 1): "f", # $[] \mapsto f$
    address(['*']): "f" # *[f] \mapsto f
  }, classic)

# In a similar way, we derive $\beta$ of shape 3.
classic = Graft(pastingDiagram(
  OpetopicInteger(3),
  {
    address([], 1): "f", # $[] \mapsto f$
    address(['*']): "f", # *[f] \mapsto f
    address(['*', '*']): "f" # [**] \mapsto f
  }, classic)

# We now take a break to derive $\omega = Y_2 \circ ([*]) Y_2$, the shape of $A$, in system OPT'.
omega = OptGraft(OptShift(OpetopicInteger(2)),
  OpetopicInteger(2),
  address([['*']]))

# Finally, we derive $A$ of shape $\omega$.
classic = Graft(pastingDiagram(
  omega,
  {
    address([], 2): "alpha", # $[] \mapsto \alpha$
    address([['*']]): "alpha" # [[*]] \mapsto f
  }, classic)

classic = Fill("beta", "A", classic)
```
CHAPTER 5

Conclusion

We have introduced two syntactic presentations for opetopes, as well as for opetopic sets, and formally related them to preexisting definitions. Together with an adequate formulation of opetopic higher categories, it is our hope that this work will be used productively for mechanical proofs of coherence in opetopic $\omega$-categories or opetopic $\omega$-groupoids.
Bibliography


APPENDIX A

Polynomial monads and the Baez–Dolan \((-\))\(^*\) construction

From a polynomial monad (definition A.1.1, corollary A.1.3, and theorem A.1.9 on pages 71–72) \(M\) with set of operations \(B\), the Baez–Dolan construction gives a new polynomial monad \(M^+\) having \(B\) as its set of colors. In this chapter, we study the monad structure of \(M^+\) in depth.

A.1. Polynomial monads

A polynomial monad is classically defined as a monoid object in \(\text{PolyEnd}(I)\), for some set \(I\). However, there are two other approaches that we present in this section. The first one presents polynomial monads as algebras over the mouthful “free polynomial monad monad” [Gambino and Kock, 2013, Kock, 2011], and the other using so-called “partial laws” [Kock et al., 2010].

Definition A.1.1 (Polynomial monad, classical definition). A polynomial monad \(M\) over a set \(I\) is a monoid object in the category \(\text{PolyEnd}(I)\) of polynomial endofunctors over \(I\), where the monoidal product \(\circ\) is the composition of polynomial functors [Gambino and Kock, 2013]. The structure morphisms \(\eta : \text{id} \rightarrow M\) and \(\mu : M \circ M \rightarrow M\) indeed give the functor \(M : \text{Set}/I \rightarrow \text{Set}/I\) the structure of a cartesian monad.

Let \(\text{PolyMnd}(I)\) be the subcategory of \(\text{PolyEnd}(I)\) spanned by polynomial monads and morphisms of monads. We emphasize that in \(\text{PolyMnd}(I)\), morphisms are expected to be identities on the set \(I\) of colors. There is an obvious forgetful functor \(U : \text{PolyMnd}(I) \rightarrow \text{PolyEnd}(I)\) that turns out to have a left adjoint that we now present. Let \(F \in \text{PolyEnd}(I)\) be as on the left, and define a new polynomial functor \(F^*\) as on the right,

\[
I \xleftarrow{s} E \xrightarrow{p} B \xrightarrow{t} I, \quad I \xleftarrow{s} \text{tr} F \xrightarrow{u} \text{tr} F \xrightarrow{r} I,
\]

where \(\text{tr} F\) is the set of \(F\)-tree (see section 2.1 on page 9), and \(\text{tr} F\) is the set of \(F\)-trees equipped with a marked leaf.

Theorem A.1.2 (Free monad [Kock et al., 2010, Kock, 2011]). The polynomial functor \(F^*\) is canonically a polynomial monad. Moreover, the adjunction \((-)^* \dashv U\) is monadic.

Let us abuse notations and write \((-)^*\) for the associated monad on the category \(\text{PolyEnd}(I)\). Write its laws by \(\mu^* : (-)^{**} \rightarrow (-)^*\) for multiplication (or unbiased grafting), and \(Y : \text{id} \rightarrow (-)^*\) for the unit (or introduction of corollas).

Corollary A.1.3 (Polynomial monad, algebraic definition). A polynomial monad is a \((-)^*\)-algebra, i.e. the data of a polynomial endofunctor \(M\) together with a morphism \(\sup : M^* \rightarrow M\) as

\[
\begin{array}{ccc}
I & \xrightarrow{\text{tr}^I M} & \text{tr} M & \xrightarrow{\text{tr}} & I \\
\downarrow & \downarrow & \downarrow & \downarrow & \\
I & \xleftarrow{\text{tr}^I} & E & \xrightarrow{t} & B & \xrightarrow{r} & I.
\end{array}
\]

(A.1.4)

such that the following two diagrams commute:

\[
\begin{array}{ccc}
M^{**} & \xrightarrow{\sup^*} & M^* \\
\mu^* & \xrightarrow{\sup} & M, \\
M^* & \xrightarrow{\sup} & M.
\end{array}
\]

(A.1.5)
The classical monad structure maps $\eta: \text{id} \to M$ and $\mu: M \otimes M \to M$ can be retrieved by noting that both $\text{id}$ and $M \otimes M$ are canonically subobjects of $M^*$. 

**Corollary A.1.6** (Contraction associativity formula). Let $T,U \in \text{tr} M$, and $[l] \in T^i$ such that the grafting $T \circ [l] U$ is defined. Then

$$t(T \circ [l] U) = t(Y_{tT} \circ_{\text{tr}[l]} Y_{tU}). \quad (A.1.7)$$

Further, for $[r] \in U^j$, we have a leaf $[l].[r] = [lr] \in (T \circ [l] U)^i$, and

$$\varphi_{T \circ [l] U}[lr] = \varphi_{V} \left( \varphi_{T[l]} \cdot \varphi_{U[r]} \right) \quad \text{with} \quad V = Y_{tT} \circ_{\text{tr}[l]} Y_{tU}. \quad (A.1.8)$$

**Proof.** Since $M$ is a $(-)^*$-algebra, the left diagram of (A.1.5) commutes. \hfill \Box

**Theorem A.1.9** (Polynomial monads via partial laws). Take a polynomial endofunctor $M \in \text{PolyEnd}(I)$, and write it as:

$$I \xrightarrow{s} E \xrightarrow{p} B \xrightarrow{t} I.$$  

A $(-)^*$-algebra structure on $M$ (i.e. a structure of polynomial monad on $M$) is equivalent to the following data:

1. (Unit) a map $\eta: I \to B$;
2. (Partial multiplication) a map $\circ: E \times I B \to B$, where for $(e,b) \in E \times I B$ and $a = p(e)$ we write $a \circ_e b$ for $\circ(e,b)$, and say that $a \circ_e b$ is an admissible expression (or just admissible); in the sequel, we assume that all expressions of this kind are admissible;
3. (Partial reindexing) for $a \circ_e b$ admissible, an isomorphism
   $$\chi_{a \circ_e b}: E(a) + E(b) - \{e\} \xrightarrow{\sim} E(a \circ_e b);$$

such that:

1. (Trivial) for $i \in I$, we have $t(\eta(i)) = i$, and $E(\eta(i))$ is a singleton whose unique element $e$ is such that $s(e) = i$;
2. (Left unit) for $i \in I$, $b \in B$, and $e$ the unique element of $E(\eta(i))$, we have $\eta(i) a_e b = b$, and $\chi_{\eta(i) a_e b} : E(b) \to E(b)$ is the identity;
3. (Right unit) for $i \in I$, $b \in B$, and $e \in E(b)$, we have $b a_e \eta(i) = b$, and $\chi_{b a_e \eta(i)}$ is given by
   $$E(b) + E(\eta(i)) - \{e\} \to E(b)$$
   $$\begin{array}{ccc}
   x & \xrightarrow{\chi_{a_e \eta}} & x \\
   y & \xrightarrow{\chi_{\eta(i)}} & e;
   \end{array}$$
4. (Disjoint multiplication) for $a \circ_e b$ and $a \circ_f c$ admissible expressions, and $e \neq f$, we have
   $$(a \circ_e b) \circ_f c = (a \circ_f c) \circ_e b,$$

and the following diagram commutes:

$$\begin{array}{ccc}
E(a) + E(b) + E(c) - \{e,f\} & \xrightarrow{\chi_{a \circ_e b}} & E(a \circ_e b) + E(c) - \{f\} \\
\downarrow & & \downarrow \chi_c \\
E(a \circ_f c) + E(b) - \{e\} & \xrightarrow{\chi_B} & E((a \circ_e b) \circ_f c);
\end{array} \quad (A.1.10)$$

with

$$B = (a \circ_e b) \circ_f c, \quad C = (a \circ_f c) \circ_e b;$$

5. (Nested multiplication) for $a \circ_e b$ and $b \circ_f c$ admissible, we have
   $$(a \circ_e b) \circ_f c = a \circ_e (b \circ_f c),$$
and the following diagram commutes:

\[
\begin{array}{c}
E(a) + E(b) + E(c) - \{e, f\} \xrightarrow{\chi_{a,b}} E(a \circ b) + E(c) - \{f\} \\
E(a \circ c) + E(b) - \{e\} \xrightarrow{\chi B} E((a \circ b)c) \\
\end{array}
\]

(A.1.11)

with

\[B = (a \circ b) \circ c.\]

Moreover with the data above, the components of the structure map \(M^* \to M\) (as in diagram (A.1.4)) are inductively given by:

1. (Target) for \(i \in I\), \(t_1 = \eta(i)\); for \(b \in B\), \(t Y_b = b\); for \(T \in \text{tr} M^*\), \([l] \in \mathcal{T}^1\), and \(b \in B\) such that the grafting \(T \circ_l Y_b\) is defined:

\[t(T \circ_l Y_b) = (tT) \circ_{\varphi_l[T]} b,\]

where \(\varphi_l\) is defined next;

2. (Readdressing) for \(b \in B\),

\[\varphi_Y : Y_b \to E(b) \] \[\{e\} \to e;\]

for \(T \in \text{tr} M^*\), \([l] \in \mathcal{T}^1\), and \(b \in B\) such that the grafting \(T \circ_l Y_b\) is defined, the readdressing \(\varphi_{T \circ_l Y_b}\) is given by

\[(T \circ_l Y_b) = T^1 + Y_b - \{[l]\} \to E((tT) \circ_{\varphi_l[T]} b)\]

\([p] \in T^1 - \{[l]\} \to \chi(tT) \circ_{\varphi_l[T]} \delta(\varphi_l[T])\]

\([e] \in Y_b \to \chi(tT) \circ_{\varphi_l[T]} \delta(e)\]

Proof (sketch). Conditions (Disjoint multiplication) and (Nested multiplication) ensure that \(tT\) does not depend on the chosen decomposition, and thus define a structure map \(sup : M^* \to M\) making the left square of (A.1.5) commute. Conditions (Trivial), (Left unit), and (Right unit) ensure that the right triangle of (A.1.5) commutes too, and thus that \(M\) is a \((-)^*\)-algebra, that is, a polynomial monad.

A.2. A COMPLETE \((-)^*\) CONSTRUCTION

Let \(M\) be a polynomial monad and define \(M^*\) as being the following polynomial functor:

\[B \xleftarrow{\varphi} \text{tr}^* M \xrightarrow{u} \text{tr} M \xrightarrow{t} B,\]  

(A.2.1)

where for \(T \in \text{tr} M\), the fiber \(u^{-1}T\) (which we shall also denote by \(T^*\)) is the set of node addresses in \(T\), and for \([p] \in T^*, s[p] := s[p] T\).

**Definition A.2.2** (Monad structure on \(M^*\)). We endow \(M^*\) with a monad structure by means of theorem A.1.9 on the preceding page. The maps \(\eta^* : I \to B\) and \(\circ : \text{tr}^* M \times_B \text{tr} M\) are defined as follows:

1. (Unit) for \(b \in B\), let \(\eta^*(b) := Y_b;\)

2. (Partial multiplication) for \(U, T \in \text{tr} M\), \([p] \in U^*\) such that \(s[p] U = tT\), write (recall that \(\circ\) is associative on the right)

\[U = X \circ Y_T \bigcirc_{\{e_i\}_i} Y_i\]  

where \(\{[e_i]\}_i \in Y_T\) (equivalently, \(\{e_i\}_i \in E(tT)\)), and define

\[U \circ T := X \circ T \bigcirc_{\{e_i\}_i} Y_i;\]
We can decompose as indeed for $U;v$ (Disjoint multiplication).

Proof of (Right unit) for $A.2.3$.

With this definition, the $(-)^*$-algebra structure map $m^*: (M^+)^* \to M^+$ of $M^*$ (see (A.1.5)) is given as follows. The target map $t: tr M^+ \to tr M$ is defined by induction: for $b \in B$, $t_b = Y_b$, for $T \in tr M$, $t_YT = T$, and for $U, v \in tr M^+$, $[l] \in \Upsilon^I$ such that $U \circ_I [l]$ is defined,

$$t(U \circ V) = (tU) \circ (tV).$$

This rest of this section is dedicated to prove that A.2.2 indeed defines a monad. We thus check the conditions of theorem A.1.9 on page 72.

Proof of (Trivial). For $b \in B$ we have $t \eta^*(b) = tY_b = b$. On the other hand, $(\eta^*(b))^* = \{\}^*$ and $s_{[l]} \eta^*(b) = s_{[l]} Y_b = b$. □

Proof of (Left unit). Let $b \in B$ and $T \in tr M$ be such that $\eta^*(b) \circ_I T$ is admissible. Then, by definition, $\eta^*(b) \circ_I T = Y_b \circ_I T = T$. Then, $\eta^*(b) \cdot T^* - \{\} = T^*$, and $\chi_{\eta^*(b) \circ_I T}$ maps $[q] \in T^*$ to $[q]$, so it is indeed the identity. □

Proof of (Right unit). Let $b \in B$, $T \in tr M$, and $[p] \in T^*$ be such that $T \circ_I [p] \eta^*(b)$ is admissible. By definition, $T \circ_I [p] \eta^*(b) = T \circ_I [p] Y_b = T$. Then, $\chi_{T \circ_I [p] \eta^*(b)}$ is given by

$$T^* + \{\} - \{[p]\} \to T^*$$

$$[p[e][p'] \in T^* \to [p] \cdot \chi^{-1}_{\eta^*(b)} e \cdot [p'] = [p[e][p']$$

$$[p'] \in T^*$$

as indeed $\chi_{\eta^*(b)}$ maps $[e] \in (Y_b)^I$ to $e \in E(b)$. □

Proof of (Disjoint multiplication). Let $A, B, C \in tr M$, $[e],[f] \in A^*$ be different, and such that $A \circ_{[e]} B$ and $A \circ_{[f]} C$ are admissible. Without loss of generality, we distinguish two cases: one where $[e] \in [f]$, for $\subseteq$ the prefix order, and one where $[e]$ and $[f]$ are $\subseteq$-incomparable.

(1) Assume $[e] \subseteq [f]$, so that $[f] = [e[q]r]$ for some $e$ and $r$, and write $A$ as (recall that $\circ$ is associative on the right)

$$A = X \circ Y_{e} \circ Y_{e[r]} \circ Y_{e[q]}.$$
where \( q \in E(tB) \) and \( \{v_i\}_i \in E(tC) \). Then
\[
A \circ B = X \circ B \circ Y \circ Y_{tC} \circ Z_i,
\]
and \( \chi_{A^{-1}}B[f] = \chi_{A^{-1}}B[e[q]r] = [e] \cdot \chi_B^{-1}q \cdot [r] \). Thus,
\[
(A \circ B) \circ C = X \circ B \circ Y \circ C \circ Z_i,
\]
and the reindexing \( \chi_{(A^{-1}B)^{-1}B} \circ \chi_{A^{-1}}B \) is given by
\[
[p] \in B^* \longmapsto [ep] \quad \rightarrow \quad [e] \cdot [rp]
\]
\[
[e[q]s] \in [f] \longmapsto [e] \cdot \chi_B^{-1}q \cdot [s] \quad \rightarrow \quad [e] \cdot \chi_B^{-1}q \cdot [s]
\]
\[
[f[v_i]s] \longmapsto [e] \cdot \chi_B^{-1}q \cdot [r] \cdot [v_i]s \quad \rightarrow \quad [e] \cdot \chi_B^{-1}q \cdot [r] \cdot \chi_C^{-1}v_i \cdot [s]
\]
On the other hand, we have
\[
A \circ C = X \circ Y_{tB} \circ Y \circ C \circ Z_i.
\]
The reindexing gives \( \chi_{A^{-1}}C[e] = [e] \), and
\[
(A \circ C) \circ B = X \circ B \circ Y \circ C \circ Z_i
\]
\[
= (A \circ B) \circ C.
\]
The reindexing \( \chi_{A^{-1}}C \circ \chi_{A^{-1}}C \) is given by
\[
[p] \in B^* \longmapsto [ep] \quad \rightarrow \quad [e] \cdot [rp]
\]
\[
[e[q]s] \in [f] \longmapsto [e] \cdot \chi_B^{-1}q \cdot [s] \quad \rightarrow \quad [e] \cdot \chi_B^{-1}q \cdot [s]
\]
\[
[f[v_i]s] \longmapsto [e] \cdot \chi_B^{-1}q \cdot [r] \cdot \chi_C^{-1}v_i \cdot [s] \quad \rightarrow \quad [e] \cdot \chi_B^{-1}q \cdot [r] \cdot \chi_C^{-1}v_i \cdot [s]
\]
We see that the square (A.1.10) commutes in the case \( [e] \in [f] \).

2. Assume \([e] \) and \([f] \) are \( \equiv \)-incomparable, and write \( A \) as
\[
A = (X \circ Y_{tB} \circ Y_i) \circ Y_{tC} \circ Z_j,
\]
Then
\[
A \circ B = (X \circ B \circ Y_i) \circ Y_{tC} \circ Z_j,
\]
the reindexing gives \( \chi_{A^{-1}}B[f] = [f] \),
\[
(A \circ B) \circ C = (X \circ B \circ Y_i) \circ C \circ Z_j,
\]
and the complete reindexing \( \chi_{(A^{-1}B)^{-1}B} \circ \chi_{A^{-1}}B \) is given by
\[
[p] \in B^* \longmapsto [ep] \quad \rightarrow \quad [e] \cdot [rp]
\]
\[
[e[q]s] \in [f] \longmapsto [e] \cdot \chi_B^{-1}q \cdot [s] \quad \rightarrow \quad [e] \cdot \chi_B^{-1}q \cdot [s]
\]
\[
[f[v_i]s] \longmapsto [e] \cdot \chi_B^{-1}q \cdot [r] \cdot \chi_C^{-1}v_i \cdot [s] \quad \rightarrow \quad [e] \cdot \chi_B^{-1}q \cdot [r] \cdot \chi_C^{-1}v_i \cdot [s]
\]
On the other hand,
\[
A \circ C = (X \circ Y_{tB} \circ Y_i) \circ C \circ Z_j,
we have \(\chi_{A^{e,f}}C[e] = [e]\),
\[
(A \circ_B C) \circ_B B = (X \circ_B Y_i) \circ_C Z_j = (A \circ_B B) \circ_C C,
\]
and further
\[
\begin{align*}
[p] \in C^* & \quad \longrightarrow \quad [p] \in B^* \quad \longrightarrow \quad [ep] \\
[e[v_i]s] & \quad \longrightarrow \quad [e[v_i]s] \quad \longrightarrow \quad [e] \cdot \chi_{B^1v_i}^j [s] \\
[f[w_j]s] & \quad \longrightarrow \quad [f] \cdot \chi_C^1w_j \cdot [s] \quad \longrightarrow \quad [f] \cdot \chi_{C^1w_j}^j [s] \\
[p] \in A^* \text{ not as above} & \quad \longrightarrow \quad [p] \quad \longrightarrow \quad [p]
\end{align*}
\]
so that the square (A.1.10) commutes in the case where \([e]\) and \([f]\) are \(\Xi\)-incomparable too.

Finally, the monad structure of \(M^*\) given in definition A.2.2 on page 73 satisfies condition (Disjoint multiplication) of theorem A.1.9 on page 72.

**Proof of (Nested multiplication).** Let \(A, B, C \in \text{tr}M, [e] \in A^*, [f] \in B^*, \) such that \(A \circ_B C \) and \(B \circ_C C \) are admissible. Write
\[
A = (X \circ_B Y_i) \quad \text{and} \quad B = Z \circ_C T_j.
\]
Then,
\[
(A \circ_B C) \circ_B B = X \circ_B Y_i,
\]
we have \(\chi_{A^{e,f}}B[f] = [ef]\), and
\[
(A \circ_B C) \circ_B C = X \circ_B (Z \circ_C T_j) \circ_C Y_i,
\]
where
\[
\alpha(\chi_{B^1v_i}^j) = \left\{ \begin{array}{ll}
[f] \cdot \chi_C^1w_j \cdot [r] & \text{if } \chi_{B^1v_i}^j \text{ of the form } [f[w_j]r], \\
\text{otherwise.}
\end{array} \right.
\]
Remark that \(\alpha(\chi_{B^1v_i}^j) = \chi_{B^{e,f}}^j C^1v_i\). The reindexing \(\chi_{(A^{e,f}B)\alpha} C \chi_{A^{e,f}B}\) is given by:
\[
\begin{align*}
[f[w_j]r] \in B^* & \quad \longrightarrow \quad [ef[w_j]r] \quad \longrightarrow \quad [ef] \cdot \chi_C^1w_j \cdot [r] \\
[p] \in B^*, [f] \notin [p] & \quad \longrightarrow \quad [efp] \\
[e[v_i]r] \in A^* & \quad \longrightarrow \quad [e] \cdot \chi_{B^1v_i}^j \cdot [r] \quad \longrightarrow \quad [e] \cdot \chi_{B^{e,f}}^j C^1v_i \cdot [r]
\end{align*}
\]
On the other hand, we have
\[
B \circ_C C = Z \circ_B Y_i \circ_C T_j,
\]
\[
(A \circ_B C) \circ_B C = (X \circ_B (Z \circ_C T_j) \circ_C Y_i) = (A \circ_B B) \circ_B C,
\]
and the reindexing is given by
\[
\begin{align*}
[p] \in C^* & \quad \longrightarrow \quad [fp] \quad \longrightarrow \quad [efp] \\
[f[w_j]r] \in B^* & \quad \longrightarrow \quad [f] \cdot \chi_C^1w_j \cdot [r] \quad \longrightarrow \quad [ef] \cdot \chi_C^1w_j \cdot [r] \\
[p] \in B^*, [f] \notin [p] & \quad \longrightarrow \quad [ep] \\
[e[v_i]r] \in A^* & \quad \longrightarrow \quad [e] \cdot \chi_{B^{e,f}}^j C^1v_i \cdot [r]
\end{align*}
\]
We thus see that the square (A.1.11) commutes, and that the monad structure of \(M^*\) given in definition A.2.2 on page 73 satisfies condition (Nested multiplication) of theorem A.1.9 on page 72.